REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
16-04-2008

2. REPORT TYPE
Final Report

3. DATES COVERED *(From – To)*
1 January 2007 - 25-Jun-09

4. TITLE AND SUBTITLE
Cell-NPE (Numerical Performance Evaluation)

5a. CONTRACT NUMBER
FA8655-07-1-3027

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. Jochem H Hauser

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
HPCC-Space GmbH
Karl-Scharfenbergstr. 55-57
Salzgitter 38229
Germany

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
Unit 4515 BOX 14
APO AE 09421

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Grant 07-3027

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking HPCC-Space GmbH as follows: B. TECHNICAL PRPOPOSA/DESCRIPTION OF WORK
Cell: A Revolutionary High Performance Computing Platform
On 29 June 2005 [1], IBM has announced that is has partnered with Mercury Computer Systems, a maker of specialized computers. The Cell chip provides massive floating-point capability and scalability for a variety of applications. It is a general-purpose processor and provides a high cost performance ratio (GFlops/$). In brief, it has the capability, because of its networking features, to provide a supercomputer in a nutshell.

This signals an important shift in the computing industry away from the traditional processor technology dominated by Intel. While in the past, the development of computing power has been driven by desktop applications; gaming, and other data-intensive applications are now driving the performance gains in computing.

A basic Cell processor is expected to deliver clock speeds of 4 GHz per core and contains nine cores, so it has about 10 times the processing power of a standard desktop PC processor. The applications that need that level of performance are mainly in the area of engineering and scientific computing.
So far pricing was not revealed, but it is believed that the Cell will cost about $30 in game consoles. The average PC processor today costs about $150 to $200. IBM has been developing the Cell in a joint venture with Sony and Toshiba since 2001. Manufacturing of the Cell started earlier this year at IBM's East Fishkill (N.Y.). The Cell processor is a radical new design. It incorporates a lot of additional number crunching and communications technology onto one chip that normally is spread among a set of chips. This produces a far more powerful package. Cell chips typically will incorporate up to nine separate cores that are programmed to handle different processing tasks. Cell offers the computing power for most demanding applications in high performance computing.
These performance achievements mean much more advanced problems can be solved. For instance, in magnetohydrodynamics, the coupling of Maxwell's equations with the nonlinear Navier-Stokes equations combined with sophisticated turbulence models in conjunction with ionized non-equilibrium flow past 3D configurations is one of the most demanding computational problems, requiring almost unlimited computing power. Using Cell processing power and advanced algorithms a quantum leap in problem solving can be expected.
IBM is in negotiation with third party manufactures for Cell high performance computing workstations. IBM Böblingen Entwicklung GmbH, Germany is currently working on a version of the Linux operating system for Cell to be released in fall 2005, and a set of tools that will make it easier for developers to build products on top of Cell.
1.1 Implementing the Numerical Test Suite to the Cell Processor

The language of choice for obvious reasons is Java, there may be some C++ code, in combination with the interace code to the Cell chip. The numerical testsuite algorithms were selected to represent the numerics encountered in typical aerospace applications or propulsion simulation. The numerical testsuite algorithms itself will not show any features that are specific to the Cell processor. Since the PPE of a Cell processor is a general-purpose computer, a Java compiler should be available soon. According to oral information from the IBM booth, AIAA Reno meeting 2006, IBM is planning to have a Java compiler for the Cell chip by end of 2006.

These challenges can only be met by most modern software technologies. Therefore, object-oriented programming (OOP) as well as the Unified Modeling Language (UML is a standard in computer science for the design of complex software) will be employed in this contract. Furthermore, only a language providing built in Internet features and security classes should be utilized. Therefore, all software will be implemented in the Java language. The authors have more than six years of experience and have performed large software projects in Java. In addition, the latest Java Development Kit (JDK) 5.0 is some 50 % faster than comparable code in C++ on a 64-bit AMD Opteron architecture. This number could even increase for the Cell processor.

1.2 Numerics

The numerics packages are described in detail in the WP description, see Sec. 2. Although not explicitly specified, code and data security are of prime importance, especially if collaborative engineering via distributed database systems on the Internet is foreseen. Java has unique security features that will be incorporated in the Cell-NPE testsuite to fully address this concern.

At present, the Java programming language together with the Java Runtime Environment is arguably the best way to obtain secure high performance computing and communications (HPCC) over the Internet. Modern Java (5.0) outperforms C++ code, for instance matrix-vector multiplication testcase (recent proposer measurements). Java was conceived with the Internet (World Wide Web) in mind and it was developed considering security issues from the very beginning. Unlike other popular programming languages (C/C++, FORTRAN etc.) or development environments overall system security was always one of Java's key features.

1.2.1 References

The references, 1-6, are on the Cell processor, namely its architecture and its compute power. They are the most recent references available on the Cell processor. Reference 7 is on the science and technology trends for the 21 first century. Reference 8 gives insight into the latest activities on grid computing. Reference 9-17 describe the state of the numerics used in multi-physics solvers. Based on this, the numerics test suite was selected.

Special reference to: Mercury Compute, IBM To Build Cell Processor-Based Systems, 29 June 2005, http://au.news.yahoo.com/050629/3/uwep.html

1. Mallinson, D., CELL: A New Platform for Digital Entertainment, Sony Computer Entertainment US Research and Development Center, Austin, TX, March 2005.
2. Blachford, N., Cell Architecture Explained, June 2005
3. Kahle, J. et al. Cell Architecture and Broadband Engine Processor, IBM Systems and Technology Group, Austin, TX, 2005
4. Unleashing the Power: A programming example of large FFTs on CELL, IBM Systems and Technology Group, Austin, TX, 2005
5. Proceedings of the GCC Developer’s Summit, June 22-24, Ottawa, Ontario, Canada, 210pp..
6. The Need for Software, Scientific Computing World, August-September 2000, Issue 54, pp.16.
7. Science and Technology Shaping the Twenty-First Century, Executive Office of the President, Office of Science and technology Policy, 1997.
8. Foster, Ian, The Grid: Computing without Bounds, Scientific American, April 2003, pp. 60-67 and Foster, Ian (ed.): The Grid: Blueprint for a new Computing Infrastructure, Morgan Kaufmann Publishers, 1999.
9. Häuser, J., Ludewig, T., Gollnick, T., Williams, R.D.: Innovative Software for HPCC. ECCOMAS 2001, Computational Fluid Dynamics Conference, Swansea, September 2001, UK
10. Häuser, J., Ludewig, T., Williams, R.D., Winkelmann R., Gollnick T., Brunett S., Muylaert J.: A Test Suite for High-Performance Parallel Java, Advances in Engineering Software, 31 (2000), 687-696, Elsevier.
11. Ginsberg, M., Häuser, J., Moreira, J.E., Morgan, R., Parsons, J.C., Wielenga, T.J.: Future Directions and Challenges for Java Implementations of Numeric-Intensive Industrial Applications, 31 (2000), 743-751, Elsevier.
12. Häuser, J., Ludewig, T., Gollnick, T., Winkelmann, R., Williams, R., D., Muylaert, J., Spel, M., A Pure Java Parallel Flow Solver, 37th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 99-0549 Reno, NV, U.S.A, 11-14 January 1999.
13. Moreira, J.E., S. P. Midkiff, M. Gupta, From Flop to Megaflop: Java for Technical Computing, IBM Research Report RC 21166.
14. Moreira, J.E., S. P. Midkiff, M. Gupta, A Comparison of Java, C/C++, and Fortran for Numerical Computing, IBM Research Report RC 21255.
15. Häuser J., Williams R.D., Strategies for Parallelizing a Navier-Stokes Code on the Intel Touchstone Machines, Int. Journal for Numerical Methods in Fluids 15, pp. 51-58., John Wiley & Sons, June 1992.
16. Winkelmann, R., Häuser J., Williams R.D, Strategies for Parallel and Numerical Scalability of CFD Codes, Comp. Meth. Appl. Mech. Engng., NH-Elsevier, 174, 433-456,1999.
17. Häuser, J., Xia, Y., Muylaert, J., Spel, M., Structured Surface Definition and Grid Generation for Complex Aerospace Configurations, In: Proceedings of the 13th AIAA Computational Fluid Dynamics Conference -Open Forum, June 29 - July 2, 1997.

C. FACILITIES/EQUIPMENT

For software development and software testing only the IBM Cell simulator is needed that has been installed on a Linux system. All test can be performed on this system. The simulator is an exact mapping of the Cell architecture so that a comprehensive testing is possible. If possible, a performance evaluation on a real Linux based Cell prototype dual core machine will be carried out. Such a machine is available at IBM Böblingen, Germany. An agreement of cooperation is pending between IBM and the proposer. It is very likely that in 2006 workstations based on the Cell chip will become available. The simulator code will run without any modifications on any Cell based computer so that any institution that has access to such a machine would be able to perform these benchmarks.

15. SUBJECT TERMS

EOARD, Aerodynamics, Computational Fluid Dynamics (CFD)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT** UL	**18, NUMBER OF PAGES** 155	**19a. NAME OF RESPONSIBLE PERSON** SURYA SURAMPUDI
a. REPORT UNCLAS	**b. ABSTRACT** UNCLAS	**c. THIS PAGE** UNCLAS			**19b. TELEPHONE NUMBER** *(Include area code)* +44 (0)1895 616021

Programming the IBM Cell Broadband Engine
a general parallelization strategy

Prepared under Grant
EOARD Grant # 073027

"Cell-NPE (Numerical Performance Evaluation)"

for
Dr. Jean-Luc Cambier
Propulsion Directorate, EAFB, CA
and
Dr. Surya Surampudi
Chief Aeronautical Science
EOARD, London

HPCC-Space GmbH
Jochem Hauser, Torsten Gollnick
Karl-Scharfenberg-Str.55,
38229 Salzgitter, Germany

April 2008

Table of Contents

Acronyms

BEI	Broadband Engine Interface
CBE	Cell Broadband Engine
DMA	Direct Memory Access
EA	Effective Address of data in the main memory of the cell system
EIB	Element Interconnect Bus
IDE	Integrated Development Environment
JDK	Java Development System
JVM	Java Virtual Machine
LBC	Lattice Boltzmann Code
LS	Local Store
MASS	Mathematical Acceleration Subsystem
MFC	Memory Flow Controller
MIC	Memory Interface Controller
OS	Operating System
PDE	Partial Differential Equation
PPE	PowerPC Processor Element
RISC	Reduced Instruction Set Computer
SDK	Software Development Kit
SIMD	Single Instruction Multiple Data
SMT	Symmetric Multi-Threading
SPE	Synergistic Processor Element
SPU	Synergistic Processor Unit (SPU is the SPE without the MFC)
XIO	XDR IO Interface (Rambus)

1. Introduction

Scientific computing and engineering simulation require enormous computing resources. The level of reality in these computations is steadily increasing. Furthermore, it is of great interest to provide small gadgets with large computing power for dedicated tasks to endowing them with special capabilities. Computing power is achieved through parallelism. At the same time, parallelism should be cost effective, too, obtained from off the shelf technology.

The Cell-chip from IBM that became very recently available promises substantial potential for high performance computing. The enormous demand for computing power in scientific and engineering applications is the target of this novel parallel architecture. It is implemented in IBM's new server blades, apart from being used in the Playstation 3. With its compact design providing highest performance, it is superior to any conventional processor that exists today. The Cell-chip reaches into the domains of today's supercomputer systems without requiring their manufacturing and maintenance costs.

The concept of the Cell Broadband Engine (CBE), a joint venture of IBM, Sony, and Toshiba is the answer to problems arising in the field of processor design. Shrinking structures (about 45 nm), higher clock rates (6 GHz and more), and architectural complexity comes along with heat production as well as communication problems at these high frequencies. These problems were addressed by the inventors of the Cell chip by reducing processor complexity and, at the same time, providing the enabling technology for computational power with multiple cores. These cores are using SIMD (Single Instruction Multiple Data) instructions for vector computations. Combining these powerful units with a general purpose processor core (IBM PowerPC technology) results in a high performance processor of reduced footprint while avoiding the enormous heat production of conventional processors.

The design of the Cell-chip enables the construction of cost effective networks of processors in comparison with supercomputer available today with concerning manufacturing cost and power consumption. Moreover, the Cell-chip is being used in the mass market for the Sony Playstation 3 and there exists the possibility of becoming a major player in the field of high performance computing.

A most interesting fact is the massively employment of the Playstation 3 in high performance computing:

A Stanford University's *Folding@Home* project, dedicated to protein folding, was getting a major boost when the Playstation 3 became available. Since the software for Folding@Home is being delivered with the Playstation 3, Stanford University faces immediately an increased amount of workload done.

The table below shows the client statistics for the Folding@Home project which is frequently updated and can be observed at:
http://fah-web.stanford.edu/cgi-bin/main.py?qtype=osstats

Client statistics by OS **Last updated at Sat, 05 May 2007 16:51:19**

OS Type	Current TFLOPS*	Active CPUs	Total CPUs
Windows	185	194188	1688072
Mac OS X/PowerPC	9	10942	98896
Mac OS X/Intel	15	4784	12452
Linux	46	27021	224799
GPU	57	970	3078
PLAYSTATION®3	583	34109	114006
Total	895	272014	2141303

Table 1: The table shows the contribution of the Playstation 3 to the computing power available to the protein folding project of Stanford University. Since the Playstation 3 is new on the market the contribution shown is remarkable.

Client statistics by OS **Last updated at Mon, 31 Mar 2008 12:03:11**

OS Type	Current TFLOPS*	Active CPUs	Total CPUs
Windows	180	189271	1971903
Mac OS X/PowerPC	7	8509	113786
Mac OS X/Intel	21	6928	43936
Linux	44	25886	281649
GPU	26	445	5291
PLAYSTATION®3	1272	42106	477372
Total	1550	273145	2893937

Table 2: The table shows the contribution of the Playstation 3 to the computing power available to the protein folding project of Stanford University 10 months later.

However, before the Cell-chip can be utilized, proper *programming and parallelizing strategies* for simulation problems in scientific and engineering computing need to be devised. The usage of the Cell processor in this field is basically different from the environment in the Playstation 3 and requires a alternative operating system to be installed. The present report *addresses these questions along with performance estimates and measurements for selected test problems* symptomatic to scientific computing. Parallel performance measurements were performed both on the Cell-chip simulator as well as on the Cell processor in form of the Playstation 3.

1.1. The Cell Processor Architecture

The novel architecture of the Cell Broadband Engine comprises two parts of processing components each requires a special approach in programming. The first part is the PowerPC Processor Element (PPE) with a dual-threaded 64-Bit PowerPC Architecture core (Reduced Instruction Set Computer, RISC) supported with a Vector/SIMD Multimedia Extension. Communication from and to the PPE takes place with its PowerPC Processor Storage Subsystem (PPSS) which has a unified 512kB level 2 instruction and data cache and is connected via the Element Interconnect Bus to IO, memory, and to **8 Synergistic Processing Elements (SPE)**. The PPE is used for driving an Operating System which must support the 8 SPEs.

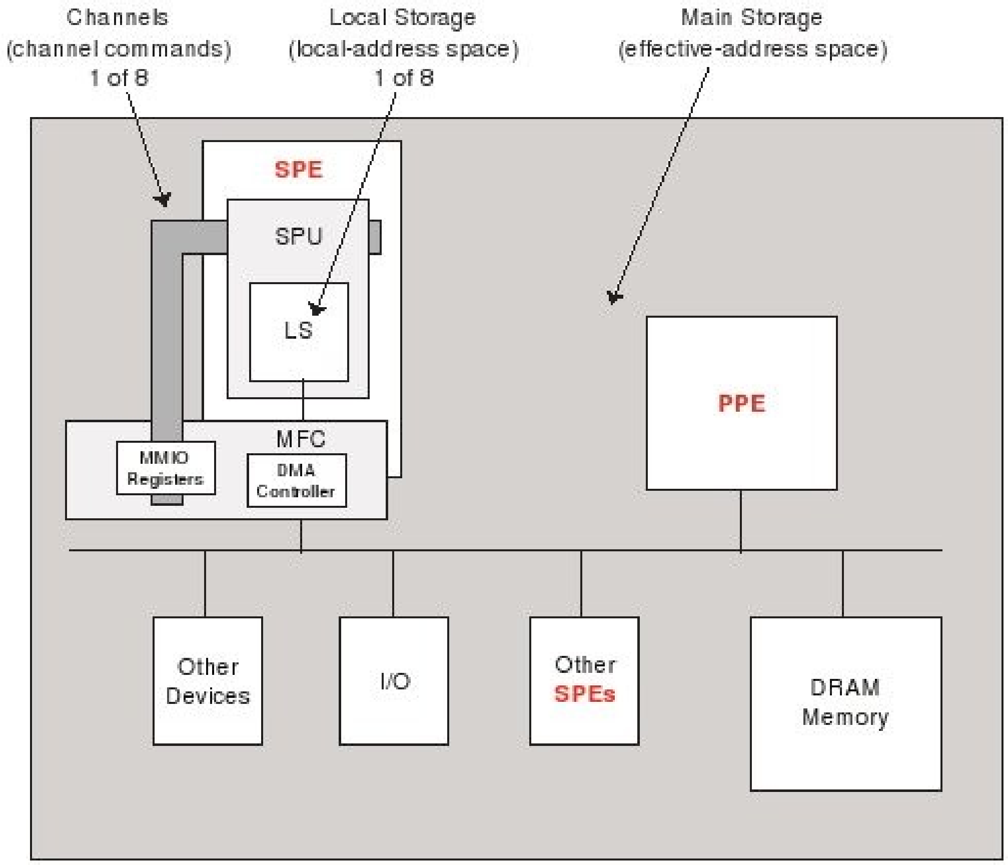


Figure 1: Depicted are the cell specific storage domains. The PPE and the SPEs can communicate with each device connected to the Element Interconnect Bus (EIB).

The SPE consists of a Synergistic Processor Unit (SPU) with a software-controlled 256-KB local store (LS) and a unified register file (128 x 128 bit), and the Memory Flow Controller (MFC). The SPU introduces its own new SIMD instruction set and works on the LS which hold both instructions and data. The access to the LS with regard to the SPU is unprotected and not managed. There is no Memory Management Unit (MMU) and no cache on the SPE. The communication between the LS and the environment (PPE / IO) of an SPE happens through DMA transfers managed by the MFC. While one DMA transfer is up to 16KB in size, the MFC can maintain DMA queues (up to 2048) and process them asynchronously while the SPU continues to work on the LS. Since DMA operations can only access the LS for at most one every eight cycles, the impact of DMA transfers is not important. The DMA transfers are coherent to main storage due to the information about virtual-memory address translation information provided by the OS to the MFC.
Both PPE an SPE have SIMD instructions though SPEs instruction set differs. The vectorization of programs might be in the duty of the programmer or can be done by the compiler.
The development kit provides C/C++ language extensions (intrinsics) for both the PPE and the SPE to gain explicit control over SIMD instructions and registers without having to apply assembler code.

1.2. High Performance Computing Model for the Cell Processor

The scientific problems intended to be solved on the Cell are grid based, that is, a solution domain is formed by an object and space surrounding it. The solution domain is then discretized forming a grid while the volumes defined by gridlines are subject to solution methods (finite volume schemes) solving nonlinear differential equations like Euler- or Navier-Stokes-equations. These continuum based problems have to be advanced in space and time in a computational space which is adapted (using curvilinear coordinate system) to computational needs. Since the cells are coupled by the influence of the computational method, the domain decomposition method to be used to distribute the computational load on the resources available must take this into account as well as the synchronization scheme between the resources. Resources are processors, multiple processor cores on hardware side as well as threads on software side.

The Cell-chip provides 8 computing cores that can be regarded as threads with fixed features. These threads are then covered by software threads by spending a layer of abstraction to be independent of the implementation of the hardware.

To be able to use the SIMD operation of the SPEs, the SPE programs as well as the data structures to be moved between the components of Cell must be modeled according to some requirements specified by the hardware.

The synchronization between the SPEs and the PPE as well as the modelling of the data structure and its positioning in the main memory is of major importance and will govern the performance results. It is to be expected that supporting libraries will be developed to de-couple the development process from hardware-centric problems.

2. Task Distribution on the CBE

The PPE being responsible for the OS is also the host for an application that is, where the main program starts up. An example discussion would be a flow solver:

The flow solver's first task is to convert a given (structured) grid into a data structure residing inside the main memory, provided the main memory is large enough. For this model structured solver we assume a finite volume scheme which lead to cells and cell faces. We view the set of cells as one domain. Depending on the kind of the desired computation a set of algorithms has to be applied on the cells and cell faces. Then the calculations takes place by employing the SPEs so that the solution evolves over time in the main memory.

2.1. The Application Partitioning Model

The SPEs can be used in different ways. They can compute the data in parallel doing the same tasks on different parts of the solution domain

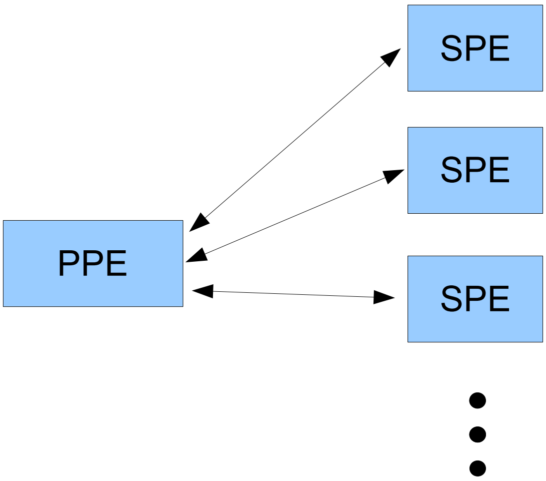
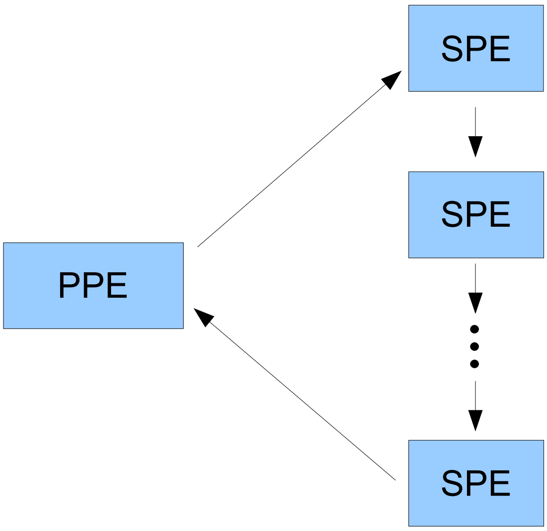


Figure 2: Depicted above is the configuration for parallel tasks working on the SPEs while each SPE has its own .communication with the PPE.

The second possibility is a multistage pipeline having the SPEs computing successive stages to advance the solution in time. One can of course model a situation where two or more parallel pipelines are employed. But this depends on the given problem.



3: This is the pipeline configuration for a set of SPEs processing different stages of the computing task. Just the tail and the head of the pipeline do interacting with the PPE.

It is a challenge to find the right task geometry of SPEs to form a computing stencil that moves over the entire solution domain with optimized communication regarding data exchange with the PPE.

The first problem we face is the decomposition of the solution domain with respect to the restriction of the SPEs. We have 256KB for the numerics and the data. Either a fixed maximum set of data with storage space for the largest numerical algorithm set is preassigned or, depending on the required amount of storage for the actual used numerics, a domain decomposition algorithm determines the maximum number of cells fitted into the remaining LS.

Another problem is how to organize the communication between the PPE and the SPE respectively the LS. *An additional note must be given that the task in the SPE runs to end. It will not be interrupted.*

There are several possibilities:

- The SPEs can be loaded with the numerics block (main program) and a list of DMA transfers that cover a certain region of the solution domain. The SPE is then responsible for exchanging data between main memory and the local store. The PPE is responsible for setting up the SPU with the next numerics block belonging to a successive list of numerical tasks (calculations on cell faces then calculations of the cell midpoints).
- The SPEs are loaded with certain parts of the solution domain and the PPE loads the numerical blocks successively so that each numerical block works on the same data. After the end of the last numerical block the data will be exchanged with the main memory.
- A mixture of the above scenarios can be found in the multistage pipeline of SPEs where one or more pipelines would be prepared with the appropriate numerical blocks. The tail of the pipeline (first SPE in the line) then must get the subdomains from main memory where the head of the pipeline would write back the advanced solution in the subdomain.

2.2. Mechanism of Communication

An efficient and versatile communication algorithm is mandatory to achieve parallel performance. Because of the small local store of an SPE, a special strategy is needed. With the present version of the Cell-chip, it is clear that performance tweaking require a **substantial programming effort**. Several programming models are presented below presented in the order of increasing complexity.

2.2.1. Global Communication

Talking about numerical blocks means that a program is compiled exclusively for the SPU (with a special compiler backend) to an object file. That is, different numerical tasks need to be organized in modules (or object files), which, of course, can be held in main memory, to be loaded into the SPE when needed.

A program (process) in the PPE can spawn a thread on an SPE, having one or more PPE-threads associated with it. These PPE threads can interact with the SPE through the LS (mapped into main memory) or indirectly through effective-address memory (addressing the LS for DMA transfers). Threads states are *poll, sleep*, and *wait*. The OS is responsible for low level handling of SPEs (scheduling of available resources to other tasks running on the PPE, runtime-loading, parameter-passing to the SPEs, event and error notification, support for debugging, and mapping the LS into the main memory)

2.2.2. Communication between PPE and SPE

Two mechanisms exist for communication between PPE and the SPEs. The first are mailboxes, working as queues for exchanging 32-bit messages. There are two mailboxes for sending messages from the SPE to the PPE and one for sending messages to the SPE. The second mechanism comprises two signal-notification channels for contacting the SPE. They can be configured for one-to-one or many-to-one signaling.

These mechanisms can be used for signaling events like updating the LS by the PPE, restoring data to main memory by the SPE after a computation, or the PPE changes the mapped LS in the main memory and then notifies the SPE for updating, etc.)

3. Parallelization Strategy for Cell Chip Based Systems

Any problem to be parallelized needs to comprise a set of elements that can be partioned. The original set of elements is called a domain or solution domain. A subdomain is termed a partition. In case PDEs are solved, which is generally the case in science and engineering applications, the solution domain is described by a numerical (discrete mesh or grid) in spacetime. Most simulation cases comprise complex geometries, for instance, simulating the flow past an entire aircraft configuration. In general, such a grid is not a Cartesian grid, but contains curvilinear coordinates. There is a widely used technique dubbed boundary conforming grids, which uses grids that are fitted to the boundary of the geometry. These grids are so called multiblock grids that is, the grid is a collection of these blocks. Each block is logically equivalent to a box that comprises a set of rectangular faces, which in turn are structured as a matrix. An entry in such a matrix is grid point, comprising in most case a triple of real numbers, denoting the coordinates of a point in physical space. This also means that neighboring relationships of individual grid points within such a block are automatically known. Each block is subsequently mapped from physical space to computational space. In computational space (CS) the resulting grid is uniform and of rectangular shape. Of course, the governing physical equations need to be transformed, too. However, the type of equation remains invariant under such a transformation. The following test cases are dealing with two dimensions (2D) and it is assumed that a collection of blocks has been generated, for instance, using recursive bisection.

In case a grid is completely unstructured, recursive bisection works as well to construct an equally sized set of partitions of the solution domain. However, the neighboring information between grid points within each partition needs to be explicitly stated (for via instance voxels), but the algorithm given below would have to be modified, since it is based on the block logic. This task is, however, outside the current study.

3.1. Case 1: Multiple Blocks on One SPE

In the following it is aassumed that the solution domain id described by a collection of blocks, i.e. by a grid which is block structured. Each block contains grid cells formed by gridlines.

If the data structure of one block fits into the available local store of the SPE, the data is loaded into the LS (Local Store), computed, and written back. This is, of course, a trivial case. In general, the data size of a block exceeds the available LS, therefore must be further partioned to fit in the local store. Communciation between these partitions is needed. The following strategy minimizes the amount of communication between main memory and LS.

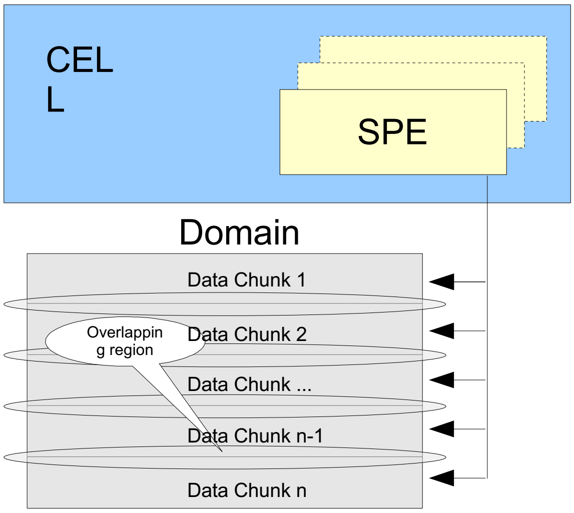


Figure 4: The monoblock domain is sliced into data chunks (rows) of equal size (the last chunk my differ in size) according to the available memory for data in the SPE's local store. works only on blockstuctured grids.

Since the logical data structure is of rectangular shape, it can be partitioned into chunks that is, multiple rows. Chunks generally are of equal size, and thus can be handled with a fixed data structure in LS. This has the following advantage: The data is organized in memory row by row, so the data handling and the alignment as demanded by the DMA process is simplified, and the load and store processes between SPE and main memory become more efficient. In addition, the data handling inside the SPE is more convenient due to row based alignment of data in memory.

Furthermore, the special property of the SPE performing DMA transfers in the background in paprllel to compuations, will be utilized by introducing a double buffering mechanism. While one chunk of data is being processed in the first buffer, the data in the second buffer is written back, and subsequently new data is loaded into this buffer.

The size of the data and the effective address of the domain will be determined in advance by the PPE and given to the SPE program at startup. The total number of data chunks depends on the available amount of memory in the LS, since the program of the SPE also resides in the LS.

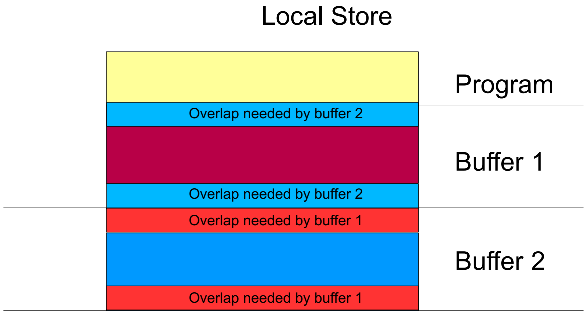

Local Store
Program
Overlap needed by buffer 2
Buffer 1
Overlap needed by buffer 2
Overlap needed by buffer 1
Buffer 2
Overlap needed by buffer 1

Figure 5: The figure shows the memory map of the LS. The program of the SPU and two buffers of equal size must fit into memory. Regions marked as overlap are needed for processing the data in the second buffer because of dependencies between the data on the buffer boundaries.

The algorithm for processing and computing data of a block by a single SPE works as follows. It should be noted that this algorithm does not foresee any communication with other SPEs that is, it works for embarrassingly parallel problems For instance, solving the Euler equations on a 2 block grid, requires additional updating between the two blocks which can be achieved though the PPE. However, the current algorithm is needed for any type of application to be implemented on the Cell-Chip.

dch_n	data chunk number n
b	buffer number
%	modulo operation

```
1. n = 1, b = 0
2. load  $dch_n$ into buffer b
3. computation of $dch_n$
   while
   loading $dch_{n+1}$ into buffer b % 1
4. computation of $dch_{n+1}$ in buffer b % 1 at the boundary to $dch_n$
   (see overlapping regions in figure 2)
5. compute $dch_{n+1}$ in buffer b % 1 at the remaining cells (but
   without those at the boundary to the next data chunk)
   while
   storing back $dch_n$ of buffer b to main memory
   while
   loading $dch_{n+2}$ into buffer b
```

```
6. compute  dch$_{n+1}$ in buffer b % 1 at the boundary to dch$_{n+2}$
7. n = n+1; b = b % 1;  goto 4
```

This algorithm does not need any external synchronization (by the PPE for instance) for processing data chunks.

In step 1, the number of chunks starts with 1, the buffer number starts with 0, subsequently alternating between 0 and 1. In step 2, the first data chunk is loaded into the first buffer. Step 3 performs the computation of the data in the first buffer, while, at the same time in the background, the second buffer is filled with the next data chunk. When the computation in the first buffer is finished, step 4 proceeds with the computation of the subsequent data chunk, but only for the common boundary region of the two buffers. It is assumed that the numerical scheme employed needs overlapping access to the data chunk in the other buffer. When this (short) computation is finished, the algorithm continues with step 5. Here the remaining data is processed while, at the same time in the background, the data of the former chunk is written back and the next chunk to be computed is read into the now empty buffer. Step 6 updates both the chunk and block number repeats the algorithm at step 4.

3.2. Case 2: One Block on Multiple SPEs

In this case the domain will be decomposed into 8 subdomains of equal size (equal size is mandatory at the boundaries) similar to the data chunks as described above. The subdomains will be assigned to SPEs and the computation of the subdomains data is the same as described in the former case. The difference to case 1 is the additional synchronization between two SPEs of neighboring subdomains. Since the decomposition into slices is the same for 2D as well as for 3D the amount of communication paths is always two (boundary update between two SPEs).

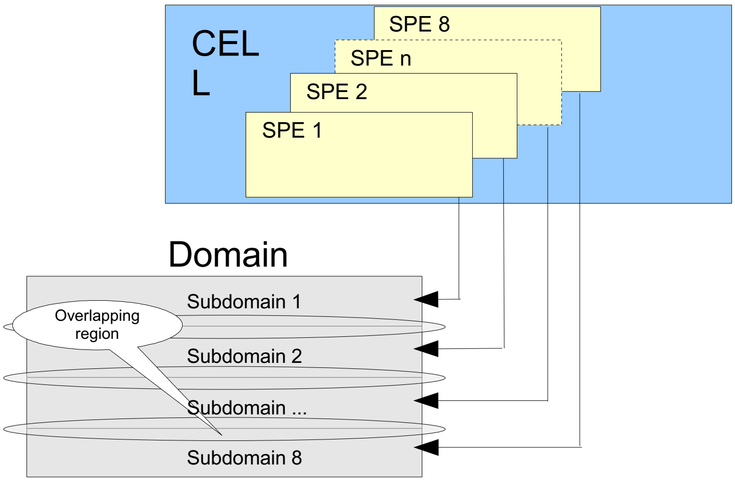


Figure 6: The picture shows the decomposition of a monoblock domain into 8 slices of equal size (the last slice may differ in size). Slices or subdomains will be assigned to SPEs for computation. There exists synchronization between two SPUs for boundary update.

This arrangement has the advantage that the PPE is not incorporated into synchronization tasks. The necessary notification for boundary updates between two SPEs can be handled by the SPEs themselves by using the mailbox mechanism or signal notification to set up a synchronization barrier. The algorithm for this case is as follows:

```
1. start programs in each SPE augmented with information about the EA and size of the subdomain as well as about neighboring SPEs for synchronization.
2. compute subdomain as described in case 1 and write back the new data to main memory.
3. write to signal notification channels of neighboring SPEs.
4. wait for neighboring SPEs to finish (blocking read of signal notification channels).
5. read boundaries of neighboring subdomains.
6. write to signal notification channels of neighboring SPEs.
7. wait for neighboring SPEs to read boundaries (blocking read of signal notification channels).
8. Goto 2.
```

It must be noted that the subdomain of an SPE includes the overlapping region of neighboring subdomains for reading. Therefore synchronization is needed to prevent the use of data of time step n+1 for a computation at time step n.

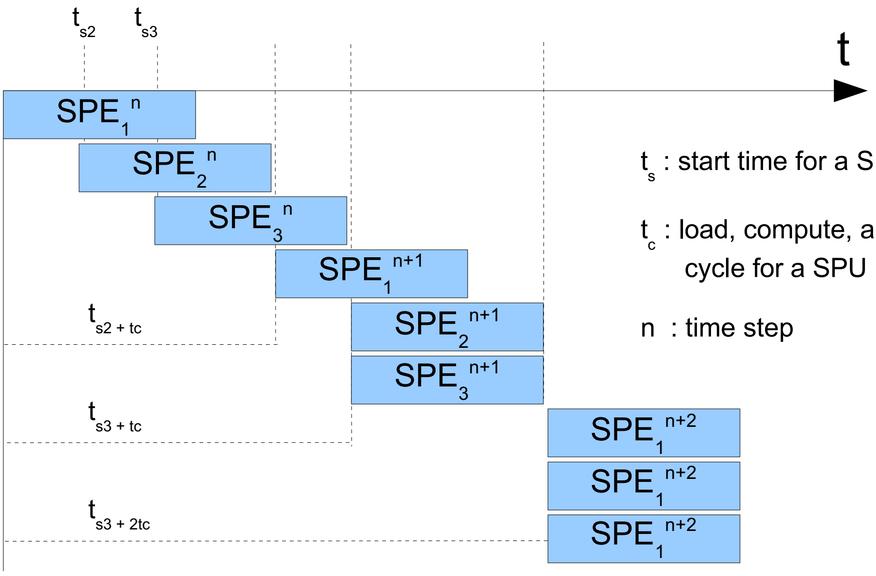


Figure 7: Time sheet for a synchronization process of an ensemble of 3 SPEs. It takes the start time of the last SPE n plus (n -1) times a load-compute-store cycle. Then normal synchronization between neighboring SPEs (subdomains) takes place. This is a rough estimate of the time the whole process to be synchronized for maximum performance.

For simplicity it is assumed that the *load-computation-store* cycle takes the same time for every SPE. This of course is not correct for regions where the numerics produces heavy load on the SPE (flow boundary layer, chemical reactions, different physical models etc.). To start the computation as soon as possible, SPEs are prepared and started sequentially by the PPE program.

3.3. *Case 3: Multiple Blocks, Multiple SPEs*

In this case each block is a domain with an additional data structure for providing overlap regions (also called ghost cells). These domains will be distributed onto the set of SPEs such that the set of domains being computed by the SPEs are connected by their joint boundaries and, if possible, the time an SPE is being active is almost the same for each SPE. This is necessary for the boundary update process to be done by the PPE. The ensemble of SPEs are processing the domains like an *advancing front,* while the boundary update thread of the PPE is always working behind, updating the regions of ghost cells.

Two different scenarios have to be considered in order to achieve load balancing (distribution of SPEs on domains)

3.3.1. The number of domains is less or equal than the number of SPEs

In this case it is assumed that the number of data blocks does not exceed the number of available SPEs. In a system with multiple Cell-chips the SPEs can be added up since the effective addresses needed for DMA transfers are available through the MFC (Memory Flow Controller).

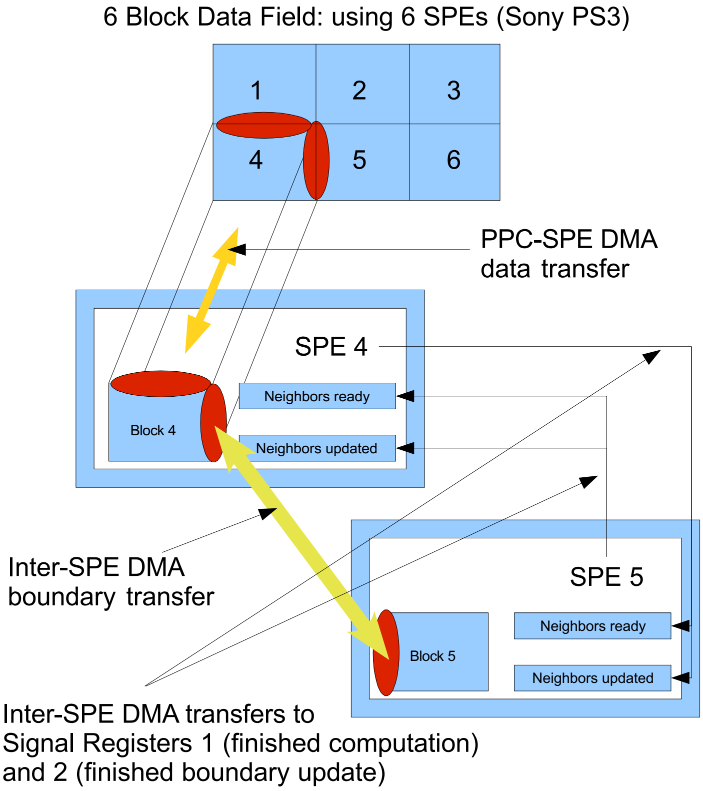

6 Block Data Field: using 6 SPEs (Sony PS3)
1
2
3
4
5
6
PPC-SPE DMA data transfer
SPE 4
Neighbors ready
Block 4
Neighbors updated
Inter-SPE DMA boundary transfer
SPE 5
Neighbors ready
Block 5
Neighbors updated
Inter-SPE DMA transfers to Signal Registers 1 (finished computation) and 2 (finished boundary update)

Figure 8: Depicted is the communication between the PPE main memory and the working SPEs regarding synchronization.

The algorithm needed for communication between neighboring SPEs takes account for the two available signaling registers belonging to every SPE. This is to achieve a decoupling of the LocalStore-MainMemory transfer process from the UpdateBoundaries process.

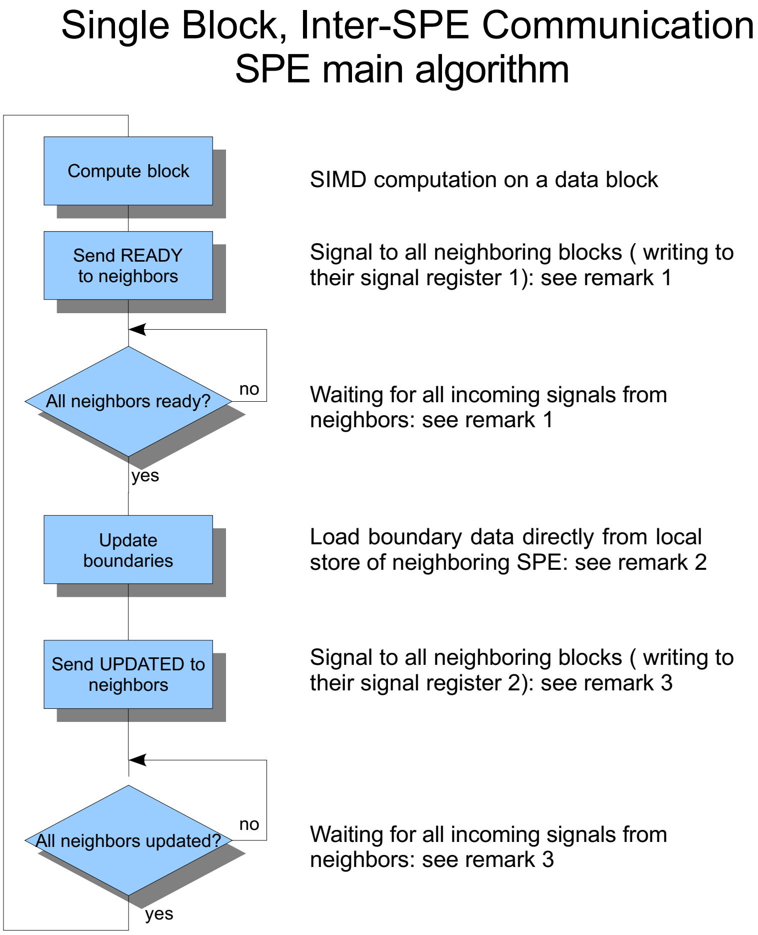

Single Block, Inter-SPE Communication
SPE main algorithm
Compute block
SIMD computation on a data block
Send READY to neighbors
Signal to all neighboring blocks (writing to their signal register 1): see remark 1
All neighbors ready?
no
yes
Waiting for all incoming signals from neighbors: see remark 1
Update boundaries
Load boundary data directly from local store of neighboring SPE: see remark 2
Send UPDATED to neighbors
Signal to all neighboring blocks (writing to their signal register 2): see remark 3
All neighbors updated?
no
yes
Waiting for all incoming signals from neighbors: see remark 3

Figure 9: The flow chart shows the algorithm used on the SPEs to compute its work packages and communicate to neighboring SPEs for updating boundaries.

Remarks on Implementing Parallel Algorithm Strategy

- Remark1

  SPEs signal that they finished compute work by writing their unique ID to signal register 1 of all neighboring blocks.#- This register is set to OR mode. IDs will be accumulated. SPE n has ID#- Writing to a register will wake up a SPE if it is in wait mode

- Remark2

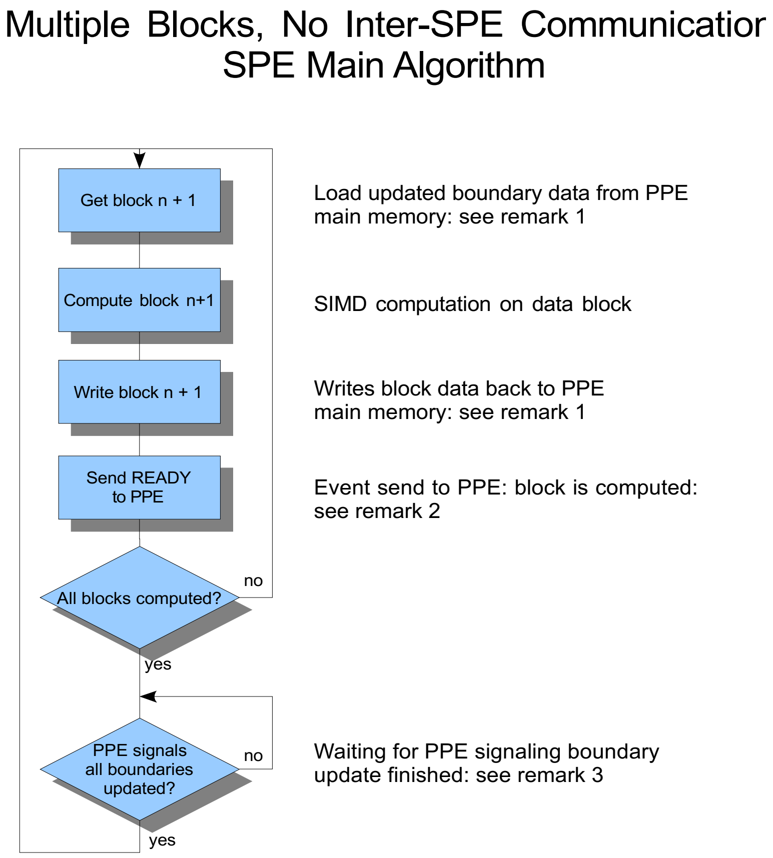
  A SPE has direct access to the local stores of its neighbors by an effective# address mapped through by its MFC (Memory Flow Controller)

- Remark3

  SPEs signal their finished update process by writing their unique ID to# signal register 2. This register is configured like register 1.

3.3.2. The number of domains is greater than the number of SPEs

Figure 10: Depicted is the communication scheme between PPE and SPEs for the number of work packages exceeding the number of SPEs and work package data exceeding the SPE local store.


Multiple Blocks, No Inter-SPE Communicatior
SPE Main Algorithm

Load updated boundary data from PPE main memory: see remark 1

SIMD computation on data block

Writes block data back to PPE main memory: see remark 1

Event send to PPE: block is computed: see remark 2

Waiting for PPE signaling boundary update finished: see remark 3

Figure 11: This flow chart shows the main algorithm on an SPE performing computations required by work packages. Here the PPE is responsible for synchronizing the boundary update process. There is no inter-SPE communication.

- Remark1

  Each block includes the boundary data (ghost cells) from its neighbors. The data transfer takes place only between PPE (main memory) and SPEs local store.

- Remark2

  Every computed block will be signaled to the PPE. This is done by an event handler that must be registered on every SPE. The event handler will activate a thread on the PPE that manages the boundary update process.

- Remark3

  The update process involves a check for blocks with fully updated boundaries. Fully updated blocks will get a signal for safely continuing computation.

The algorithm for an efficient distribution of SPEs on the domains by involving a thread of the PPE for processing the boundary updates is currently being developed.

4. The Software Development Kit (SDK) 2.0

4.1. Installation

IBM provides an installation guide describing the steps to be done. It is available on http://www.alphaworks.ibm.com/topics/cell as well as the SDK itself with additional information.

4.1.1. Prerequisites

The dedicated machine should have the following properties:

- A processor not slower than 2GHz,
- 1 GB of main memory,
- and at least 10 GB of disk space.

The supported operating system is *Fedora Core5*. This must be installed prior to the SDK.

The developer packages should be installed. This is an option of the Fedora package installer.

A working network connection to the Internet must by available.

Then additional packages, not installed automatically by the OS installer, must be added to the system. This can be done by applying the following commands as root:

```
yum install tcl
yum install tk
yum install freeglut
```

The installation procedure is described in the installation manual delivered by IBM.

Proceed as follows:

1. As root, download the development kit's ISO disk image: *CellSDK20.iso.*
2. As root, install all pre-requisites as documented in the Installation Guide.
3. Create a mount directory, such as */mnt/cellsdk*, as follows: `mkdir -p /mnt/cellsdk`
4. Mount the disk image on the mount directory, as follows:
   `mount -o loop CellSDK20.iso /mnt/cellsdk`
5. Change directory (`cd`) to /mnt/cellsdk/software.
6. Install the package by using the following command: `./cellsdk install.`
7. Change directory (`cd`) to any directory that is not the mount directory and is not below it.
8. Unmount the disk image as follows: `umount /mnt/cellsdk`

The command “`cellsdk install`” connects to a server of the Barcelona Supercomputer Center and loads all necessary packages not provided by the ISO image. It installs the files into the directories */opt/cell, /opt/ibm, and /opt/ibmcmp*.

4.2. Update: The Software development Kit 3.0

This is now integrated in the Redhat package system and conveniently installable into the Redhat Fedora Core 7 Linux distribution.

4.3. Utilization and Operating Sytems

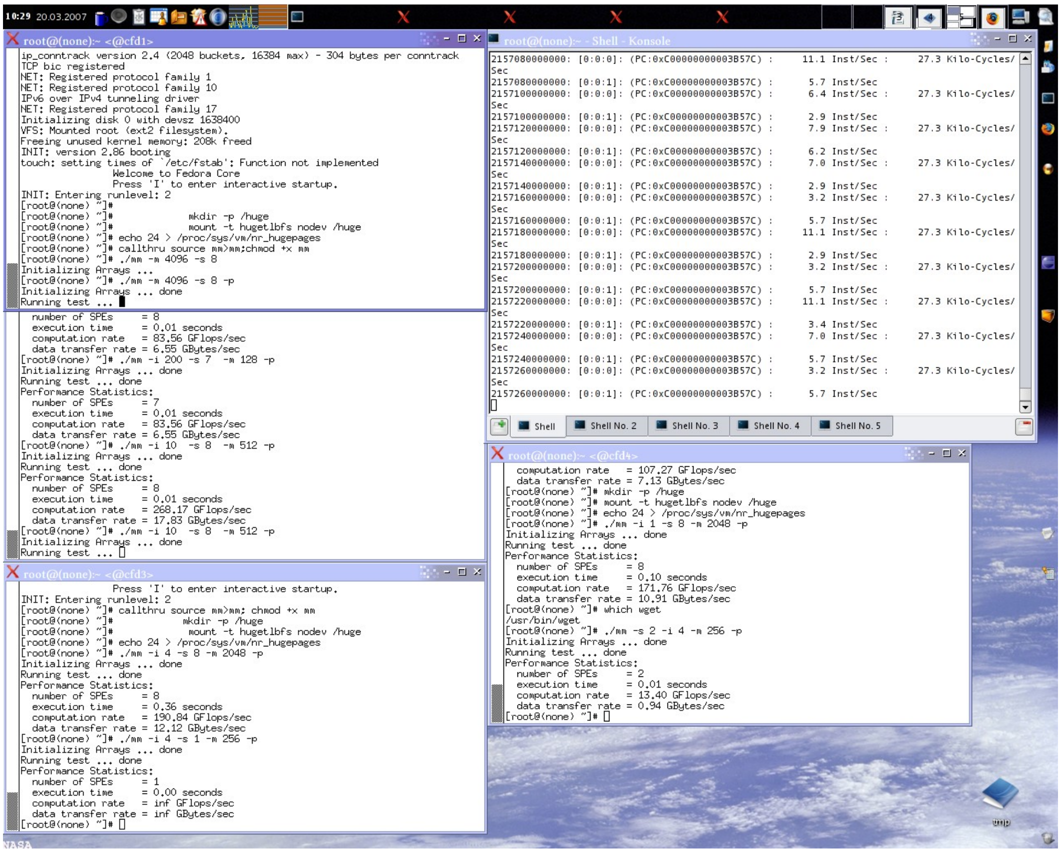
The startup script for a command shell (here we use bash) should add the following paths to binaries:

```
export PATH=$PATH:/opt/ibm/cell-sdk/prototype/bin:/opt/ibm/
systemsim-cell/bin:/opt/cell/toolchain-3.3/bin:
```

The Cell BE SDK is not only usable on Fedora Core 5. We tested that it is also possible to copy the installation directories to a different computer running Ubuntu "**Edgy Eft**" (with kernel 2.6.17) or Ubuntu "**Feisty Fawn**" (with kernel 2.6.20). Other OS may be usable, but were not tested.

The installation used was stored on a network shared folder that was accessible to all nodes in this network. The Cell SDK was installed for 32-Bit and 64-Bit (*amd64*) architectures and could therefore be used with 32- and 64-Bit linux operating systems. Depicted below is a snapshot of a desktop showing 4 Cell-simulators running on a cluster of 4 nodes. It might be possible to interconnect the simulated environments to a working simulated Cell BE cluster.

Before starting the simulator one has to copy the file "



```
ip_conntrack version 2.4 (2048 buckets, 16384 max) - 304 bytes per conntrack
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
Initializing disk 0 with devsz 1638400
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 208k freed
INIT: version 2.86 booting
touch: setting times of `/etc/fstab': Function not implemented
                Welcome to Fedora Core
                Press 'I' to enter interactive startup.
INIT: Entering runlevel: 2
[root@(none) ~]#
[root@(none) ~]#             mkdir -p /huge
[root@(none) ~]#             mount -t hugetlbfs nodev /huge
[root@(none) ~]# echo 24 > /proc/sys/vm/nr_hugepages
[root@(none) ~]# callthru source mm>mm;chmod +x mm
[root@(none) ~]# ./mm -m 4096 -s 8
Initializing Arrays ...
[root@(none) ~]# ./mm -m 4096 -s 8 -p
Initializing Arrays ... done
Running test ... 
```

```
  number of SPEs     = 8
  execution time     = 0.01 seconds
  computation rate   = 83.56 GFlops/sec
  data transfer rate = 6.55 GBytes/sec
[root@(none) ~]# ./mm -i 200 -s 7  -m 128 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 7
  execution time     = 0.01 seconds
  computation rate   = 83.56 GFlops/sec
  data transfer rate = 6.55 GBytes/sec
[root@(none) ~]# ./mm -i 10  -s 8  -m 512 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 8
  execution time     = 0.01 seconds
  computation rate   = 268.17 GFlops/sec
  data transfer rate = 17.83 GBytes/sec
[root@(none) ~]# ./mm -i 10  -s 8  -m 512 -p
Initializing Arrays ... done
Running test ... 
```

```
                Press 'I' to enter interactive startup.
INIT: Entering runlevel: 2
[root@(none) ~]# callthru source mm>mm; chmod +x mm
[root@(none) ~]#            mkdir -p /huge
[root@(none) ~]#             mount -t hugetlbfs nodev /huge
[root@(none) ~]# echo 24 > /proc/sys/vm/nr_hugepages
[root@(none) ~]# ./mm -i 4 -s 8 -m 2048 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 8
  execution time     = 0.36 seconds
  computation rate   = 190.84 GFlops/sec
  data transfer rate = 12.12 GBytes/sec
[root@(none) ~]# ./mm -i 4 -s 1 -m 256 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 1
  execution time     = 0.00 seconds
  computation rate   = inf GFlops/sec
  data transfer rate = inf GBytes/sec
[root@(none) ~]# 
```

```
2157080000000: [0:0:0]: (PC:0xC00000000003B57C) :     11.1 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157080000000: [0:0:1]: (PC:0xC00000000003B57C) :      5.7 Inst/Sec
2157100000000: [0:0:0]: (PC:0xC00000000003B57C) :      6.4 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157100000000: [0:0:1]: (PC:0xC00000000003B57C) :      2.9 Inst/Sec
2157120000000: [0:0:0]: (PC:0xC00000000003B57C) :      7.9 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157120000000: [0:0:1]: (PC:0xC00000000003B57C) :      6.2 Inst/Sec
2157140000000: [0:0:0]: (PC:0xC00000000003B57C) :      7.0 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157140000000: [0:0:1]: (PC:0xC00000000003B57C) :      2.9 Inst/Sec
2157160000000: [0:0:0]: (PC:0xC00000000003B57C) :      3.2 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157160000000: [0:0:1]: (PC:0xC00000000003B57C) :      5.7 Inst/Sec
2157180000000: [0:0:0]: (PC:0xC00000000003B57C) :     11.1 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157180000000: [0:0:1]: (PC:0xC00000000003B57C) :      2.9 Inst/Sec
2157200000000: [0:0:0]: (PC:0xC00000000003B57C) :      3.2 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157200000000: [0:0:1]: (PC:0xC00000000003B57C) :      5.7 Inst/Sec
2157220000000: [0:0:0]: (PC:0xC00000000003B57C) :     11.1 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157220000000: [0:0:1]: (PC:0xC00000000003B57C) :      3.4 Inst/Sec
2157240000000: [0:0:0]: (PC:0xC00000000003B57C) :      7.0 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157240000000: [0:0:1]: (PC:0xC00000000003B57C) :      5.7 Inst/Sec
2157260000000: [0:0:0]: (PC:0xC00000000003B57C) :      3.2 Inst/Sec :     27.3 Kilo-Cycles/
Sec
2157260000000: [0:0:1]: (PC:0xC00000000003B57C) :      5.7 Inst/Sec
```

```
  computation rate   = 107.27 GFlops/sec
  data transfer rate = 7.13 GBytes/sec
[root@(none) ~]# mkdir -p /huge
[root@(none) ~]# mount -t hugetlbfs nodev /huge
[root@(none) ~]# echo 24 > /proc/sys/vm/nr_hugepages
[root@(none) ~]# ./mm -i 1 -s 8 -m 2048 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 8
  execution time     = 0.10 seconds
  computation rate   = 171.76 GFlops/sec
  data transfer rate = 10.91 GBytes/sec
[root@(none) ~]# which wget
/usr/bin/wget
[root@(none) ~]# ./mm -s 2 -i 4 -m 256 -p
Initializing Arrays ... done
Running test ... done
Performance Statistics:
  number of SPEs     = 2
  execution time     = 0.01 seconds
  computation rate   = 13.40 GFlops/sec
  data transfer rate = 0.94 GBytes/sec
[root@(none) ~]# 
```

Figure 12: Shown above is a snapshot of the desktop showing 4 Cell-simulators running on a PC-cluster.

5. The Software Development Kit

5.1. SDK 2.1

The next implementation of the SDK comes with a few differences in the core libraries as well as modifications in the simulator. Apart from updated and improved components, the new SDK introduces support for the upcoming enhanced CBEA-compliant processor which will drastically accelerate double-precision floating point operations on the SPEs (fully pipelined, double-precision).

Especially the SPE runtime management library experienced a major change in the function set supporting the SPE:

The SPE-thread construct defined by the former library is exchanged by a data structure called SPE context, which contains low-level information about a SPE and its program. To have the thread mechanism implemented, the programmer uses a standard POSIX thread library to control the execution of the code. This provides a flexible approach to multicore programming when using the main code on different multicore architectures.

5.2. SDK 3.0

This version of the SDK provides a major update to stability and functionality. Aside from delivering new supporting libraries like

- ALF (Accelerator Library Framework): simplifies data and task management
- BLAS (Basic Linear Algebra Library)
- DaCS Data Communication and Synchronization Library

the GNU compiler suite now supports Fortran and Ada.

The IDE extension to the Eclipse development environment was updated.

6. Playstation 3: Linux and Cell SDK 2.1/3.0

To be able to use the Cell SDK 2.1 on the Playstation 3 the Linux operating system *Fedora Core 6* must be installed. Following the procedure described by Nicholas Blachford (www.ps3coderz.com) the installation will be straightforward. A TFT-Monitor with HDMI input, a USB mouse, and a USB keyboard are mandatory. After setting up the network one can access the PS3 by terminal (using ssh secure socket layer) without the needs for external components.

The SDK 3.0 requires the installation of Fedora Core 7. The installation is described by documents provided by IBM. The SDK packages will be managed by the package management of the operating system.

7. Sample Codes and Performance Tests

The tests described in the following are mostly embarrassingly parallel and do not require special communication between SPEs in general. Due to the programming effort needed for the Cell-chip they were selected to be programmed as first examples.

7.1. Theoretical Performance

The SPU provides instructions like **multiply** and **add** a vector (4 x 32 Bit) per cycle. This is also provided by a library as an intrinsic for the programmer (*spu_madd* see C/C++ Language Extension for Cell Broadband Engine Architecture). Using 8 SPUs we have

$$8 \times 8\,SPE \times 3.2GHz = 204.8\,GFLOPS$$

plus 25 GFLOPS provided by the PPE.

7.2. Prerequisites

To be able to obtain reasonable information from the Cell BE system emulator, it has to be specifically prepared. Since the simulator can be cycle accurate, the SPUs must be set into this special mode with a command provided by the simulator: `set_spu_model_pipeline` on the command shell or by using. The simulator also provides a fast mode, which is not usable for performance tests and should not be enabled.

As a more realistic testbed, the Playstation 3 with 6 SPEs was used as test platform.

7.3. Mandelbrot Set

A Mandelbrot set is computed by iterating the equation $z = z^2 + c$ with z and c being complex numbers. The computations starts with $z = 0$ and c containing the point to be evaluated. If the modulus of z does not exceed 2 , the point computed is a member of the Mandelbrot set.

The code for the PPE and the SPEs is *not yet vectorized*, and thus shows a *substantially reduced* performance. In the next stage, the vectorized code would be compared against the present version, to assess the impact of vectorization on performance.

Processor architecture	***Number of SPEs***	***Resolution/ Pixel***	***Time/s***
Cell 3.2 GHz	1	501 x 400	1.2924
Cell 3.2 GHz	2	501 x 400	0.6505
Cell 3.2 GHz	3	501 x 400	0.7694
Cell 3.2 GHz	4	501 x 400	0.5217
Cell 3.2 GHz	5	501 x 400	0.5037
Cell 3.2 GHz	6	501 x 400	0.3917
Cell 3.2 GHz	7	501 x 400	0.3647
Cell 3.2 GHz	8	501 x 400	0.3131
Cell 3.2 GHz	0 = PPE is used	501 x 400	0.2269
AMD64, 2.4 Ghz, 1 MB cache	0	501 x 400	0.1285
AMD Opteron 252, 2.6 Ghz, 1 MB cache	0	501 x 400	0.1176

Table 3: *The table shows that this problem requires better load balance for the SPEs. The first two entries of the table show a speedup of two, since the problem could be symmetrically distributed on two SPEs. This symmetry is no longer available if more SPEs are used.*

Since the SPEs do compute different problem sizes (cut along the *x*-axis), the computational load produced is not big enough to achieve maximum performance. When two instead of one SPE are used, performance doubles. But with increasing number of SPEs, the program *does not scale linearly*. This is due to the nonlinear load produced by the Mandelbrot algorithm. As a result, domain decomposition must be done in a different way, which is a **major topic for porting algorithms to the Cell BE.**

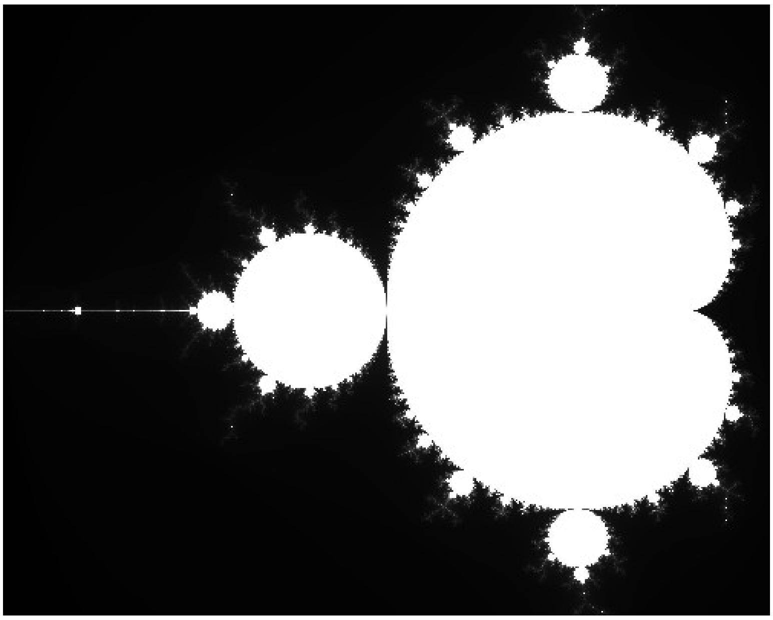


Figure 13: The data for this picture was generated by the Mandelbrot-program running on the CBE-Full System Simulator. A conversion tool was used to convert the data to a jpeg file.

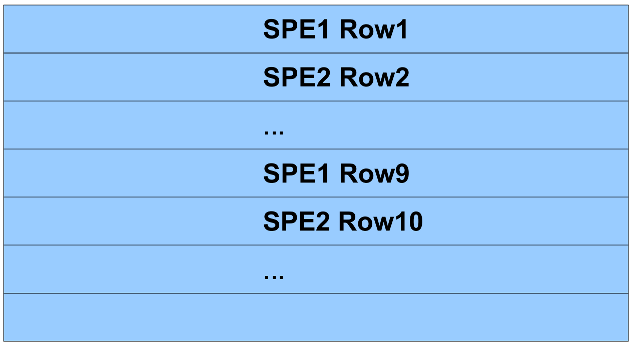
The performance of an Athlon AMD64 4000+ (1MB cache) with 2.4 GHz is superior to the performance of the PPE. This is to be expected, and is due to the missing effective branch prediction and prefetching on the PPE (as well as on the SPE).

The true performance of the Cell will be shown as soon as the computation is vectorized on all components of the Cell. The current non-vectorized Mandelbrot source code can be found in Appendix I, section 13.1

7.4. Gaussian Elimination

In this sample code, a matrix is inverted using Gaussian elimination. Gaussian elimination without pivoting is designed as follows. For test purposes (parallel strategy) only a matrix of size 481 x 481 was chosen.

The rows of this matrix were distributed in the following manner: SPE1 gets rows 1 and 9, SPE2 gets rows 2 and 9,... etc. This scattering of rows provides optimal load for the SPEs.



SPE1 Row1
SPE2 Row2
...
SPE1 Row9
SPE2 Row10
...

Figure 14: Rows are scattered on the SPEs.

Since the algorithm walks down the rows for computing sub-rows the total number of SPEs used must be adapted to the size of the problem. This avoids that an SPEs becomes idle.

The current version of the Gaussian elimination on the Cell chip is able to calculating the row echelon form of a given matrix. The values for number of SPEs, number of rows per SPE and the size of the matrix must be inserted into the source code.

A global synchronization must be implemented to take care about the following:

- The row with its left side brought to zero must be distributed to all SPEs. this can be done a a mailbox notification or by writing the row back to main memory and reading it into the SPEs local stores.
- Finally the SPEs have to write back their rows into the main memory.

In fact both mailbox mechanism and signal notification are used in the current implementation to satisfy the synchronization needs.

7.4.1. Simulation Results Gaussian Elimination

Computations were carried out on the Playstation 3 (PS3), which supports 6 SPEs under the Linux operating system (one SPE is switched off for higher yield, and one is used for the hypervisor which guards the PS3 hardware.

For the inversion of a dense square matrix of size 481 (each SPE gets the same amount of rows and the first one is not altered by the algorithm) the system delivers about **800 MFLOPS**. This is **roughly 4 times the performance of an Athlon XP Barton** (512k cache, 2.4GHz) using a sequential code.

This result is not satisfactory. To measure the performance of the inner core of the SPE program the decrementor facility of the SPE is used in conjunction with switching off synchronization.

The result is almost 11 times better (**about 8.6 GFLOPS**). This clearly indicates *that the PPE should not be involved in order to achieve synchronization* between SPEs.

7.4.2. Lessons learned

The synchronization mechanism is found to be expensive since all SPEs must report to the PPE. It will be better to distribute the actual rows for computing directly to all other SPEs.

The core algorithm is implemented using SIMD technique by utilizing SPU intrinsics provided by an IBM library. This accelerates the computation but requires some sort of alignment of the data.

Additionally the arrangement of data in the local store must fit better to SIMD operations.

The size of the matrix is limited by the size of the local store of the SPE. A mechanism must be developed to have each SPEs loading its proper rows from the main memory and writing back temporary solutions.

Data transfer is done by DMA operations. This requires to have the data aligned to 2, 4, 8, 16 or multiple of 16 Bytes because of the restrictions of the DMA mechanism. Optimal performance can be achieved by having source space and target space be aligned to 128 Bytes. The code is using a strict alignment for data transfers.

Any misalignment will cause a bus error which stops the program.

The source code can be found in Appendix I in section 13.2

7.5. Matrix Multiplication

Two matrix multiplication programs were tested. The first program written for matrix multiplication uses scalar operations. The program runs on one SPE. With a square matrix of size 60, the performance is 253 MFLOPS. This is normal since SIMD-operations are not used in the code, but they are used internally by the SPE (see Appendix I section 13.3).

The second program, also running on one SPE, uses SIMD operations together with a slightly adapted data structure to support the SIMD mechanism. The performance for the matrix of size 140 is 7.25 GFLOPS (see Appendix I section 13.4). The performance depends also on the problem size. The table below shows performance data for matrices of different size.

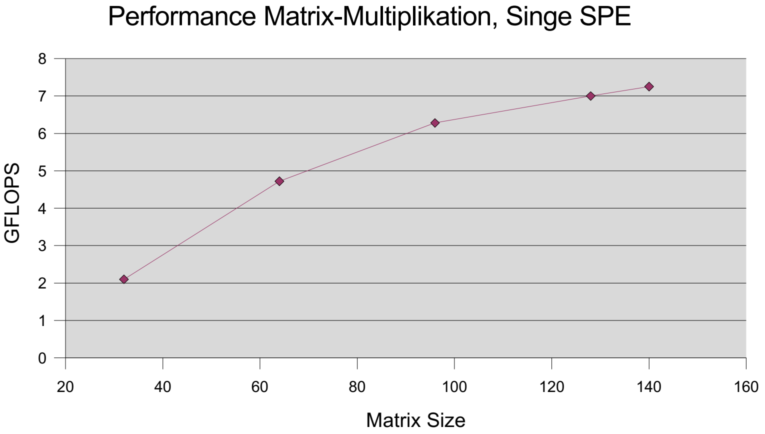
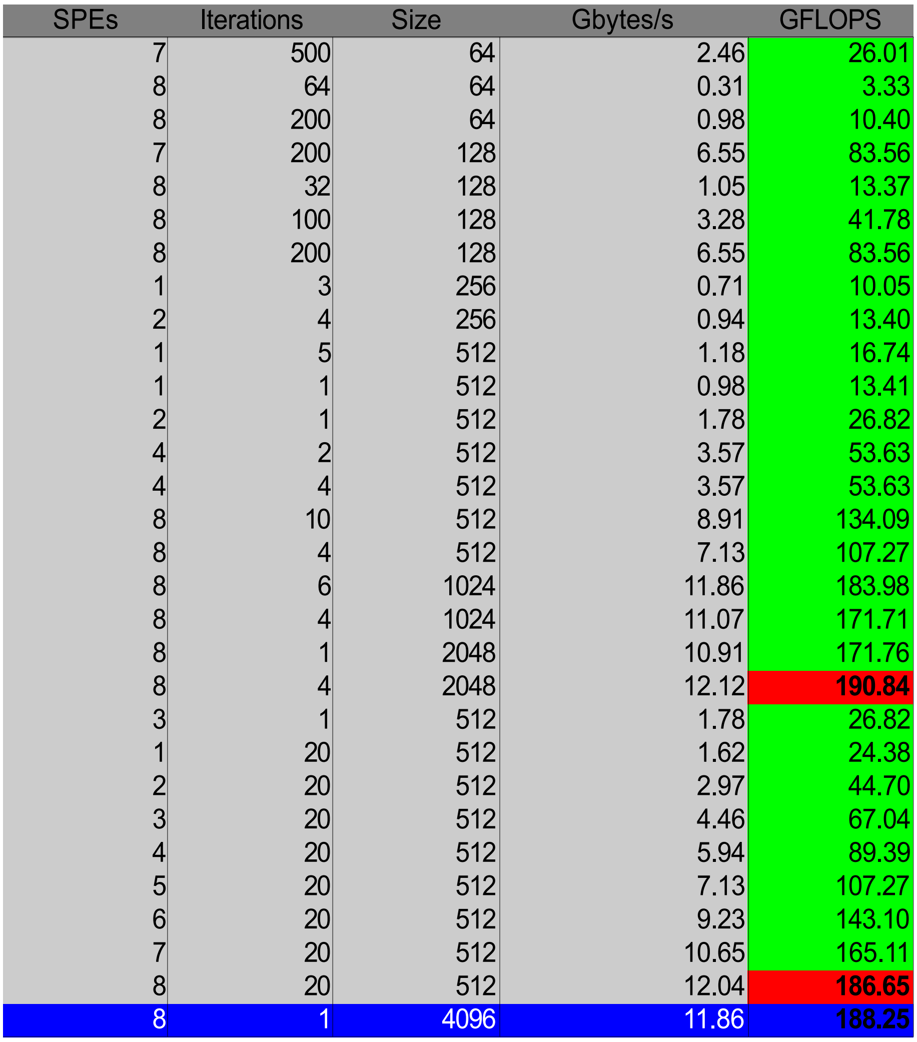


Figure 15: The chart shows the performance of one SPE processing matrices of different size for our own code. The performance increases with increasing matrix size. Towards the maximum matrix size of 140 x 140 the slope of the performance curve decreases, possibly due to memory alignment problems. The final performance value has not yet been determined, but is most likely not much higher than indicated in the figure.

The data size needed in the local store of the SPE is 128 x 128 x 4 (float) x 3 (matrices A, B, and C).

A sophisticated matrix-multiplication was developed by IBM and is part of the Cell SDK distribution. It is highly optimized and runs on a dual cell system that can be emulated by the Cell simulator (it did, however, not run on the Playstation3 so far). Depending on problem size the simulator takes a very long time running this task. For the largest test-case with a square matrix of size 4096, the simulator was running for about 2 days. This code is extremely optimized and the actually achieved performance is close to the theoretical performance of 200 GFlops of the Cell chip. On the other hand, the code is complex and about 20 times larger than the normal matrix-matrix multiplication program.

The table with the measured performance results for different numbers of SPEs and different problem sizes is depicted below.



SPEs	Iterations	Size	Gbytes/s	GFLOPS
7	500	64	2.46	26.01
8	64	64	0.31	3.33
8	200	64	0.98	10.40
7	200	128	6.55	83.56
8	32	128	1.05	13.37
8	100	128	3.28	41.78
8	200	128	6.55	83.56
1	3	256	0.71	10.05
2	4	256	0.94	13.40
1	5	512	1.18	16.74
1	1	512	0.98	13.41
2	1	512	1.78	26.82
4	2	512	3.57	53.63
4	4	512	3.57	53.63
8	10	512	8.91	134.09
8	4	512	7.13	107.27
8	6	1024	11.86	183.98
8	4	1024	11.07	171.71
8	1	2048	10.91	171.76
8	4	2048	12.12	**190.84**
3	1	512	1.78	26.82
1	20	512	1.62	24.38
2	20	512	2.97	44.70
3	20	512	4.46	67.04
4	20	512	5.94	89.39
5	20	512	7.13	107.27
6	20	512	9.23	143.10
7	20	512	10.65	165.11
8	20	512	12.04	**186.65**
8	1	4096	11.86	**188.25**

Figure 16: Performance results for matrix-matrix multiplication based on simulator computations. The blue row shows the performance of the Cell processor for matrix-matrix multiplication for a matrix of size 4096 x 4096. This value is close to the peak performance of 200 Gflops.

In order to obtain a performance close to peak value, either the size of the matrix must be large enough (4096 x 4096) or the number of repetitions must be chosen appropriately, see column 2 of Fig. (5), to produce sufficient load on the Cell processor. The multiplication for the 4096 x 4096 matrix took about 2 days to finish see last row of Fig. (5). This is due to the fact that the simulator was running in cycle-accurate mode to get reliable performance information. The source code can be found in Appendix I section 13.5.

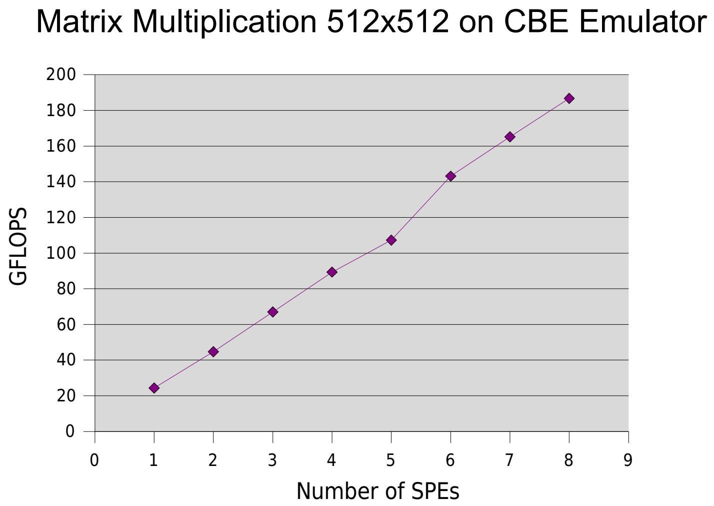


Figure 17: These test-cases were performed with a matrix of size 512 and different numbers of SPEs. There were 20 multiplications for each run. The code shows a **nearly linear speedup up to peak performance.**

Another matrix-multiplication program was investigated, namely a code programmed by Daniel Hackenberg (Technical University of Dresden, Germany) that also runs on a double Cell system but uses a different approach than the IBM code. The software computing core is written in assembler and performs better than the IBM code. It shows 401 GFlops on 2 Cell-chips (see ref. 26). Of course, this code is also extremely optimized and *accepts only matrices of dimension of multiples of 128*.

7.6. Breadth-First-Algorithm (BFS)

This algorithm is used for searching a graph and is mostly used in problems regarding computer graphics, astrophysics, genomics, and the like. It is simple to implement on a mono-processor system, but needs an enormous effort to be implemented on the Cell-chip. The Applied Computer Science Group of the Pacific Northwest National Laboratory (Fabrizio Petrini, Daniele Scarpazza, Oreste Villa) implemented this algorithm on the Cell BE (see Document Sources 17 and 25). Cell chip performance in comparison to other architectures is shown in Fig. (4), taken from a talk by F. Petrini at the *Summit on Software and Algorithms for the Cell Processor* (see Document Sources 17 and 25).

The details of the implementation were given in an article, published by Dr. Dobbs on 9 March 2007 (see ref. 25).

An interesting detail given in this article is the ratio of performance gained to programming effort.

Prozessor	Performance edges/ second	Lines of code
P4 Hyperthreading 3.4 Ghz	2.40×10^7	60
Cell BE 3.2 Ghz	5.38×10^8	1,200

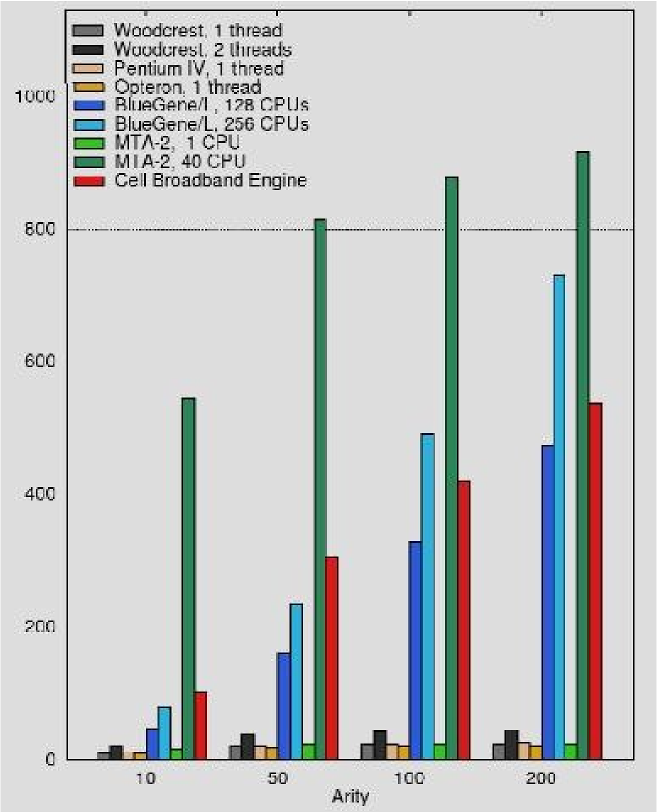


Figure 18: The performance test shows the search speed in million edges per second versus arity (gives the maximum number of children of the root and internal nodes of the graph tree). This picture is taken from the talk “Challenges in Mapping Graph Exploration Algorithms on Advanced Multi-core Processors” given by Fabrizio Petrini at the Summit on Software and Algorithms for the Cell Processor (see Document Sources 17. and 25)

The picture shown above shows that the Cell-chip can be compared to supercomputer architectures like the Blue Gene Series of IBM. Conventional processors cannot achieve this high performance.

In addition, the cost to performance ratio of the Cell can be by far better than it is the case for conventional supercomputer systems, provided the Cell-chip can effectively be employed in multi-processor configurations.

8. Best Practice and Lessons Learned

As the results from the *Matrix-Matrix* multiplication have shown, the IBM package provided almost peak performance, namely 190 Gflops out of 200 Gflops (on the simulator only). This is a very impressive number, but we did not succeed in running this code on the Playstation 3. This needs to be investigated further. On the other hand, when analyzing the source code of this example, a large amount of programming effort was detected. The code is approximately 20 times (see Appendix I) larger than the simple matrix-matrix multiplication code and needs fine tuning even on the bit level. Our own, much simplified version, obtained 56 GFlops on 8 SPEs.

The usual way to program the Cell BE is first to port a given algorithm to the PPE. After the parallelization of the algorithm the SPEs must be utilized which requires special care.

The problem size dedicated to the SPEs must be designed with the size of the local store (256 kByte) in mind. Since algorithms are also stored in the local store, they should be prepared to be loaded on demand only into the local store to reserve as much memory as possible to the data structures. Present experience can be summarized as follows:

- Data structures utilized must be aligned to a cache line (128 Bit) and DMA transfers should be initiated by the *individual SPE* (each can enqueue 16 DMA requests) rather than by the PPE, which can only enqueue 8 DMA requests.
- SPEs *must* be programmed with vector operations (exploiting SIMD) and loops should be unrolled.

A more detailed description on how to achieve highest performance out of the Cell BE chip is given at http://www-128.ibm.com/developerworks/power/library/pa-celltips1/ by Daniel A. Brokenshire (brokensh@us.ibm.com), Senior Technical Staff Member, IBM STI Design Center. However, this assumes working on the assembler level, something not feasible for the scientist or engineer who is interested in solving his problem of interest. Clearly, there must be an intermediate level that takes care of the low level programming. Perhaps this can be achieved with the concept published in ref. 26.

9. The new Cell-Chip

As already supported by the updated Cell-SDK 2.1 the upcoming next version of the Cell-chip will be used in the future Los Alamos National Laboratory's (LANL) Peta-FLOP computer Roadrunner (A 65 nm SPE for a 1 petaflop super computer, Brian Flachs, Ph.D., SPE Microarchitect, IBM Systems and Technology Group, SCEI/Sony Toshiba IBM (STI) Design Center, Austin Texas), which increase double precision of the Cell chip from 35.% Gflops to 102 Gflops. In addition, the present maximum of 3 Gbyte of addressable XDR memory will be boosted to 16 Gbytes using conventional DDR2 memoryly supports Conclusions and Future Work

10. Conclusions and Future Work

The current study has shown that the IBM Cell-Processor provides an *enormous potential* (an application can reach some 200 Gflops, or in the case of protein folding a combined 583 TeraFlops were already achieved), but at the cost of a very sophisticated programming model, which is most likely beyond the level of the average engineer or scientist who wishes to solve and run his specific application. Perhaps this issue will be addressed by IBM, and a compiler that alleviates the data distribution for parallelization and also helps with vectorization can be expected soon. Second, the next generation of Cell-processors will have a floating point performance that will match their integer performance. A Java compiler could/should also be provided. So far, only oral statements by IBM employees were heard. *The Java thread concept would be most helpful in code parallelization.*

The current conclusion is: ***programming the Cell processor requires substantially more programming effort in order to get high performance.*** *Code on the Cell processor can easily be 20 times larger than the conventional counterpart, possessing extreme complexity.*

Research should therefore be focused on devising a high level programming model, allowing to use an extremely powerful and versatile language like Java. Second, sample code reflecting realistic applications in science and engineering should be implemented on the Cell chip, and performance as well parallelization speedup should be measured, in particular for complex 2D and 3D geometries.

In the following, research activities are listed that should be performed as a next step. It should be understood that each topic itself is a major piece of work, but obtaining the proper goal would also be a major step forward on the path of to off the shelf high performance computing.

1. *To develop strategies for alleviating the programming of the cell chip,*
1. *Achieving the aim of producing software prototypes for realistic science and engineering applications ,*
2. *To improve raw performance for Cell chip applications,*
3. *To build clusters of Cell chips,*
4. *Investigating other forthcoming novel processor architectures like the SUN Rock chip (2008) or a quad (octet) AMD chip etc.*

Here, we only briefy allude how these goals could be reached.

1. There is a recent paper by Noll et al. that presents a runtime environment (RE) implementing a Java Virtual Machine (JVM) for the Cell chip. This JVM co-executes between the different SPEs and provides an automatic software based memory management system for instruction and data caching. This RE should be implemented and tests should be performed uitilizing the applications that are of interest in simulating science and engineering. In particular, it should be investigated how the Java language is supported.
2. While problems treated so far were embarrassingly parallel, problems requiring communication across domain boundaries will be considered, which is normally the case in simulating PDEs. Since PDEs occur in almost all fields of engineering and science, this is arguably the most important class of problems. To this end , a **Laplace problem** should be programmed on all 8 SPEs, to demonstrate the mapping of complex geometries onto the Cell processor. This is important, because many of the real applications are described by complex 2D or 3D geometries, like turbine blades, complete turbines, multiple airfoils, aircraft configurations, or nozzles geometries etc. Furthermore, parallelization strategies presented will be refined. Extensive performance data need to be computed.
3. It should also be investigated how a different class of simulation codes will perform on the Cell-chip, namely Lattice Boltzmann Codes (or Direct Simulation Monte Carlo). A simple LBC code can be written and tested on a PC. In the next stage, this code should be ported to the Cell-Chip, and extensive performance and parallel scaling measurements should be carried out.
4. In addition, the Playstation 3 was already used in this report to provide performance numbers. It is planned to purchase a second Playstation 3 computer, in order to investigate parallelism, vectorization, and overall performance, eventually linking two Playstation 3 computers to measure combined parallel performance. At present, Playstation 3 comprises 256 MB of memory only in the longer run, building a cluster from Playstation 3 computers should be foreseen, and the implementation of realistic simulation software in the area of MHD (magneto-hydrodynamics) should be envisaged.
5. In 2008 new processors will come out. Software design and parallelization strategies in concert with the above mentioned sample codes should be implemented on these processors, in order to quickly perform comparisons between the Cell chip and the new architectures.

11. Competition in HPC : GPU (Graphics Processing Units)

Since the development of graphics cards is strongly coupled with the development of computer games, the demand for computational power is still increasing, especially when realistic physical behavior of game objects is needed. More and more computational aspects of games are moved from the CPU to the GPU. To overcome these computational loads graphics card developers decided to parallelize their hardware to have the data of scenes and dynamic behavior of game objects computed as fast as possible. In the following the properties of cards using NVIDIAs GPUs G80,G92 are described:

In general a GPU is constructed through multiple scalar processors (up to 128 and more) which can, as a group, work on local cache to minimize the slower data transfer with the GPUs main memory (located on the graphics card). The connection to the card's memory is from 64 bit, 6.4 GByte/s (low end 8400GS) to 512 bit, 128GByte/s (high end 9800GX2).

The graphics cards do have main memory from 256Mbyte to 1.5Gbyte. The connection to the elements on the motherboard is done with PCI-Express 2.0 which gives 512Mbytes/s per lane. It is common to have a x16 slot (16 x 512Mbyte/s = 8Gb/s) for the cards. A connection with x32 (16Gbytes/s) is possible but not available yet.

The number of cards used is restricted by the motherboards capabilities. Up to four cards a are possible right now in the consumer space (i.e. Intel X48 motherboards).

Special versions of cards dedicated for HPC are available (NVIDIA Tesla series).

The programming language used is C with special extensions. NVIDIA is providing a C language development environment called CUDA (Compute Unified Device Architecture)

The GPU operates as a coprocessor to the CPU and is viewed as a device executing a high number of threads in parallel. Threads can be organized in blocks which can efficiently operating on fast shared memory and be synchronized as a group. Thread blocks can be organized in a grid so that the graphic card , depending on its capability, can execute on block of threads in parallel or multiple blocks of a grid sequentially.

The scalar processors of the GPU are grouped into so called multiprocessors which have a SIMD architecture and on-chip memory.

Because of the **much easier software implementation** and the availability, for instance, of the **CUDA library,** scientific-technical high performance computing currently **should focus on the usage of the NVIDIA graphics cards**. Therefore, research should be started and/or continued using this hardware. In particular the extreme performance / price ratio of the NVIDIA cards is highly attractive.

12. Document Sources (URLs)

An intensive search for useful documents about programming the Cell processor was undertaken. Documents that were found useful, are listed below.

1. An introduction to the IDE for the Cell Broadband Engine SDK
http://www-128.ibm.com/developerworks/edu/pa-dw-pa-cellide.html?S_TACT=105AGX59&S_CMP=GR&ca=dgr-lnxw02aIDECellBESDK

2. Barcelona Supercomputer Center (BSC)

http://www.bsc.es/projects/deepcomputing/linuxoncell/?S_TACT=105AGX16&S_CMP=DWPA

3. CELL: A New Platform for Digital Entertainment
http://www.research.scea.com/research/html/CellGDC05/index.html

4. Cell (microprocessor) - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Cell_(microprocessor)

5. Cell Broadband Engine Architecture and its first implementation
http://www-128.ibm.com/developerworks/power/library/pa-cellperf/

6. Cell Processor Net
http://www.cell-processor.net/news.php

7. Cell architecture forum
http://www-128.ibm.com/developerworks/forums/dw_forum.jsp?forum=739&cat=46

8. Cell Architecture Explained
http://www.blachford.info/computer/Cell/Cell0_v2.html

9. CellPerformance.com
http://www.cellperformance.com

10. HPC Consortium -- Optimization of the Power architecture for a multiplicity of disciplines.
http://www.hpc-consortium.net/

11. Help - IBM Education Assistant
http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp

12. IBM Research | IBM Research | Compiler Technology for Scalable Architectures
http://domino.research.ibm.com/comm/research_projects.nsf/pages/cellcompiler.index.html

13. Open Platform for PLAYSTATION®3
http://www.playstation.com/ps3-openplatform/index.html

14. Power.org - Cell Developer Corner
http://www.power.org/resources/devcorner/cellcorner/

15. Programming high-performance applications on the Cell BE processor, Part 1: An introduction to Linux on the PlayStation 3
http://www-128.ibm.com/developerworks/power/library/pa-linuxps3-1/?ca=dgr-lnxw07Linux-and-PlayStation%203

16. Sony Computer Entertainment Inc.
http://cell.scei.co.jp/e_download.html

17. Workshop Software & Alg on Cell
http://www.cs.utk.edu/~dongarra/cell2006/

18. The Cell project at IBM Research
http://www.research.ibm.com/cell/

19. developerWorks : Power Architecture technology
http://www-130.ibm.com/developerworks/power

20. louiscandell.com - HOWTO - PlayStation 3 - Install Ubuntu Linux / Debian Linux on the PS3
http://www.louiscandell.com/ps3/

21. 6.189 Multicore Programming Primer
http://cag.csail.mit.edu/ps3/index.shtml

22. Exploiting Single Precision Arithmetic and Achieving Full Precision Accuracy
http://icl.cs.utk.edu/iter-ref/

23. Cell Broadband Engine technology
http://www.alphaworks.ibm.com/topics/cell

24. BLOG Cell Broadband Engine
http://cellbe.blogspot.com/

25. Dr. Dobb's | Programming the Cell Processor | March 9, 2007
http://ddj.com/dept/cpp/197801624;jsessionid=TPMOF3UBISZF2QSNDLOSKHSCJUNN2JVN

26. Noll. A. et al. : **CellVM: A Homogeneous Virtual Machine Runtime System for a Heterogeneous Single-Chip Multiprocessor, preprint, 11 pp.**

27. NVIDIA CUDA: preview
Triolet, Damien, , www.behardware.com, March 21, 2007

28. CUDA Programming Guide
NVIDIA,Version 1.1 November 29,2007

28. General-Purpose Computation Using Graphics Hardware
www.gpgpu.org

13. Appendix I: Source Code Listings

13.1. Mandelbrot Set

```
/*
 * =====================================================================================
 *
 *        Filename:  cellMandel.c
 *
 *     Description:  This program computes a Mandelbrot set
 *
 *         Version:  0.1.0
 *         Created:  06/23/2006 04:37:26 AM EDT
 *        Revision:  see svn
 *        Compiler:  ppu32-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  ©HPCC-Space GmbH
 *
 * =====================================================================================
 */
```

```
#include <stdio.h>
#include <ctype.h>
#include <libspe.h>
#include <sched.h>
#include <stdlib.h>
#include <libmisc.h>
#include <limits.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <fcntl.h>

//#include <malloc_align.h>


#include "cellMandel.h"

extern spe_program_handle_t worker;

/** main function.
 *
 */
double yMin = -1.0;
double yMax = +1.0;
double xMin = -2.0;
double xMax = +0.5;

double dxy = 0.05;

u_int8_t *res;




void usage(char * name)
{
  printf("USAGE: %s [options]\n", name);
  printf("       Valid Options include:\n");
  printf("         -s #   : perform with specified number of SPEs. Default is 0.\n");
  printf("                :  0 SPEs means perfom just on PPE.\n");
  printf("         -d #   : step width of domain. Default is 0.05.\n");
  exit(1);
}

void getTime(int print)
{
  static struct timeval t1; /* var for previous time stamp */
  static struct timeval t2; /* var of current time stamp */
  struct timezone tzp;
  if(gettimeofday(&t2, &tzp) == -1)
    exit(0);
  if(print == 1)
  {
    double elapsed_seconds=(double)(t2.tv_sec - t1.tv_sec) + ((double)(t2.tv_usec - t1.tv_usec))/1000000.0;
    printf("Time spent [%.4fs] \n", elapsed_seconds);
  }
  t1 = t2;
}


int
main (int argc, char *argv[])
{
  double cx, cy;
  double zx, zy, new_zx;
  unsigned char n;
  int nx = 0, ny = 0, i = 0;
  int counterN = 0;
  int numOfSPE = 0;
  int datasize = 0;
  int stop = 0;

  for (i = 0; i < argc; i++)
  {
    if (*argv[i] == '-')
    {
      switch (*(argv[i]+1))
      {
      case 's':
        i++;
        if (i < argc && isdigit(*argv[i]))
        {
          numOfSPE = atoi(argv[i]);
          if (numOfSPE < 0 || numOfSPE > 8)
            numOfSPE = 0;
        }
        else
        {
          printf("ERROR: Number of SPEs is not specified.\n");
          usage(argv[0]);
        }
        break;
      case 'd':
        i++;
        if (i < argc && isdigit(*argv[i]))
        {
          dxy = atof(argv[i]);
          if (dxy < 0)
            dxy = 0.05;
        }
        else
        {
          printf("ERROR: step width is not specified.\n");
          usage(argv[0]);
        }
        break;
      case 'n':

        stop = 1;
        break;

      }
    }
  }

  for (cx = xMin; cx < xMax; cx += dxy)
  {
    nx++;

  }
  printf("nx = %i\n",nx);
  for (cy = yMin; cy < yMax; cy += dxy)
  {
    ny++;

  }
  printf("ny = %i\n",ny);
  datasize = nx * ny * sizeof(u_int8_t);

  printf("datasize = %i\n",datasize);

  control_block cb[8]__attribute__ ((aligned (128))) ;

  getTime(0);

  if(numOfSPE == 0 && !stop)
  {
    res =  (u_int8_t *) malloc(datasize * sizeof(u_int8_t));


    for (cy = yMin; cy < yMax; cy += dxy)
    {
      for (cx = xMin; cx < xMax; cx += dxy)
      {
        zx = 0.0;
        zy = 0.0;
        n = 0;
        while ((zx*zx + zy*zy < 4.0) && (n != UCHAR_MAX))
        {
          new_zx = zx*zx - zy*zy + cx;
          zy = 2.0*zx*zy + cy;
          zx = new_zx;
          n++;
        }
        res[counterN++] = n;

      }
    }
  }
  else
  {

    if (DATASIZE < datasize/numOfSPE)
    {
      perror("Datasize is too large for one DMA transfer\n");
      exit (1);
    }
    speid_t spe_ids[numOfSPE];


    int nySPE = ny / numOfSPE;

    int nySPErem = nySPE + (ny - nySPE * numOfSPE);

    printf("nySPE = %i\n", nySPE);
    printf("nySPErem = %i\n", nySPErem);

    if (stop)
      exit(0);
    int countSPE = 0;


    float yMaxtmp = yMax;

    for ( countSPE = 0; countSPE < numOfSPE-1; countSPE++)
    {
      yMaxtmp = yMin +  (nySPE -1) * dxy;
      //      cb[countSPE].data = malloc_align(nx * nySPE * sizeof(u_int8_t),7);
      cb[countSPE].data = malloc_align(DATASIZE,7);
      cb[countSPE].xMin = xMin;
      cb[countSPE].xMax = xMax;
      cb[countSPE].yMin = yMin;
      cb[countSPE].yMax = yMaxtmp;
      cb[countSPE].dxy = dxy;
      cb[countSPE].ny = nySPE;
      printf("----------\n");
      printf("yMin = %f\n",yMin);
      printf("yMaxtmp = %f\n",yMaxtmp);

      spe_ids[countSPE] = spe_create_thread(0, &worker, &cb[countSPE], NULL, -1, 0);
      if ( (int)spe_ids[countSPE]  == -1)
      {
        perror("Unable to create SPE thread");
        return (1);
      }

      yMin = yMaxtmp + dxy;

    }

    nySPE = nySPErem;

    cb[countSPE].data = malloc_align(DATASIZE,7);
    yMaxtmp = yMin +  nySPE * dxy;


    cb[countSPE].xMin = xMin;
    cb[countSPE].xMax = xMax;
    cb[countSPE].yMin = yMin;
    cb[countSPE].yMax = yMaxtmp;
    cb[countSPE].dxy = dxy;
    cb[countSPE].ny = nySPE;
    printf("-----------\n");
    printf("yMin = %f\n",yMin);
    printf("yMaxtmp = %f\n",yMaxtmp);


    spe_ids[countSPE] = spe_create_thread(0, &worker, &cb[countSPE], NULL, -1, 0);
    if ( (int)spe_ids[countSPE]  == -1)
    {
      perror("Unable to create SPE thread");
      return (1);
    }

    int status;
    for (countSPE = 0; countSPE < numOfSPE; countSPE++)
    {
      (void) spe_wait(spe_ids[countSPE], &status, 0);
    }
  }

  getTime(1);

  //
  printf ("To process the image: convert -depth 8 -size %dx%d gray:picture pic.jpg\n",
          nx, ny);
  FILE *fp = fopen("picture","wb");
  if (fp != NULL)
  {
    if (numOfSPE == 0)
      fwrite(res, (size_t)sizeof(u_int8_t), (size_t)counterN, fp);
    else
    {
      for (counterN = 0; counterN < numOfSPE; counterN++)
        fwrite(cb[counterN].data , (size_t)sizeof(u_int8_t), nx * cb[counterN].ny, fp);

    }

    fclose(fp);

  }
  return 0;
}

/*
 * =====================================================================================
 *
 *        Filename:  worker.c
 *
 *     Description: Mandelbrot set computation on the SPE
 *
 *         Version:  0.1.0
 *         Created:  06/23/2006 07:46:11 AM EDT
 *        Revision:  none
 *        Compiler:  spu-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  ©HPCC-Space GmbH
 *
 * =====================================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spu_intrinsics.h>
#include <spu_mfcio.h>
#include <sys/types.h>
#include <malloc_align.h>
#include "cellMandel.h"

volatile control_block cb;
volatile unsigned char *res;

int
main (unsigned long long spe_id, unsigned long long parm)
{
  float cx,cy,zx,zy,new_zx;
  unsigned int counterN;
  unsigned char n;
  unsigned int tag_id = 0;
  spu_writech(MFC_WrTagMask, -1);
  unsigned int i;

  spu_mfcdma32( (void *) (&cb), (unsigned int) parm, sizeof(control_block),tag_id, MFC_GET_CMD);

  (void) spu_mfcstat(MFC_TAG_UPDATE_ALL);
  printf("Got CB\n");

  int iter = 0;
   for (cy = cb.yMin; cy <= cb.yMax; cy += cb.dxy)
    {
      for (cx = cb.xMin; cx <= cb.xMax; cx += cb.dxy)
      {
        zx = 0.0;
        zy = 0.0;
        n = 0;
        while ((zx*zx + zy*zy < 4.0) && (n != UCHAR_MAX))
        {
          new_zx = zx*zx - zy*zy + cx;
          zy = 2.0*zx*zy + cy;
          zx = new_zx;
          n++;
        }
        res[counterN++] = n;
        iter += n;
      }
    }

  printf("Number of iterations = %i\n",iter);

  for ( i = 0; i < DATASIZE/MAXDSIZE; i++)
  {
    spu_mfcdma32((void *)(res + i * MAXDSIZE), (unsigned int) (cb.data + i * MAXDSIZE), MAXDSIZE, tag_id, MFC_PUT_CMD);
  }

  (void) spu_mfcstat(MFC_TAG_UPDATE_ALL);
  printf("Data written\n");
  return 0;
}
```

```
#ifndef CELLMANDEL_H_
#define CELLMANDEL_H_


#include <stdlib.h>
#define DATASIZE 229376
#define MAXDSIZE 16384

typedef struct _control_block
{
  float xMin;
  float xMax;
  float yMin;
  float yMax;
  float dxy;
  unsigned char *data;            /* address to be filled by single-buffered DMA */
  int ny;
  int pad;
}
control_block ;

#endif /*CELLMANDEL_H_*/
```

13.2. Gaussian Elimination (under development)

```
/*
 * =====================================================================================
 *
 *        Filename:  cellGaussElim.c
 *
 *     Description:  This program performs Gauss elimination
 *
 *         Version:  0.9.1
 *        Revision:  see svn
 *        Compiler:  ppu32-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  ©HPCC-Space GmbH
 *
 * =====================================================================================
 */


#include <stdio.h>
#include <libspe2.h>
#include <sched.h>
#include <stdlib.h>
#include <libmisc.h>
#include <time.h>
#include <sys/time.h>
#include <pthread.h>

#include "cellGaussElim.h"

//#define PLOT


#define ARR(uv,j,i) *((uv) + (j) * ra9nk + (i))
```

```
/**
 * References a SPE program linked to this executable (CESOF)
 */
extern spe_program_handle_t worker;


typedef struct thread_data
{
  spe_context_ptr_t spe_context;
  pthread_t ppe_thread;
  spe_sig_notify_1_area_t *sig_notify_ps_area;
  spe_spu_control_area_t *spe_spu_control_area;
  unsigned int entry;
  void *argp;

}
thread_data_t;


// startup the thread on the SPE
void *ppe_pthread_function(void *arg)
{
  thread_data_t *threadData = (thread_data_t *) arg;
  if (spe_context_run(threadData->spe_context, &threadData->entry, 0,threadData->argp,NULL,NULL) < 0)
  {
    perror("Failed running context");
    exit(1);
  }

  pthread_exit(NULL);
}


int
main (void)
{
  int n = 481;
  int i,j;
  int numOfSPUs = 6;
  int rowsPerSPU = 80;

  thread_data_t threads[numOfSPUs];

  // setup the array ----------------------------------------------------------------------

  float *arrayA[DMA128(n)];

  for ( i = 0; i < n; i++)
  {
    // one slot per row for making DMA transfers easy
    arrayA[i] = malloc_align(DMA128(n) * sizeof(float), 7);
    for ( j = 0; j < n; j++)
      arrayA[i][j] = rand() % 5 +1;

  }
  for ( i = 0 ; i < n; i++)
    arrayA[i][i] = 10;

  //setup of SPEs ----------------------------------------------------------------------------
  //getTime(0);
  control_block cb[numOfSPUs] __attribute__ ((aligned (128)));

  for (i = 0; i < numOfSPUs; i++)
  {
    cb[i].numOfRows = rowsPerSPU;
    cb[i].arrayA = arrayA;
    cb[i].size = n;
    // set alignement according to the number of rows per SPE

    cb[i].rowIDs = malloc_align(DMA128(rowsPerSPU) * sizeof(int), 7);
    for (j = 0; j < rowsPerSPU; j++)
    {

      cb[i].rowIDs[j] = i + 1 + j * numOfSPUs;
    }


    // creating the SPE context
    if ((threads[i].spe_context = spe_context_create(SPE_MAP_PS,NULL)) == NULL)
    {
      perror("Failed creating context");
      exit(1);
    }

    // load the SPE binary into the SPE context
    if (spe_program_load(threads[i].spe_context, &worker))
    {
      perror("Failed loading program");
      exit(1);
    }

    // init
    threads[i].entry = SPE_DEFAULT_ENTRY;
    threads[i].argp = &cb[i];

    // run the thread
    if(pthread_create(&threads[i].ppe_thread, NULL, &ppe_pthread_function, &threads[i]))
    {
      perror("Failed creating thread");
      exit(1);
    }

    //get the signal stat  prof_stop();e area of the SPE
    if ((threads[i].sig_notify_ps_area = spe_ps_area_get(threads[i].spe_context, SPE_SIG_NOTIFY_1_AREA)) == NULL)
    {
      perror(" Failed getting spe's signal state area");
    }
    //get the control state area of the SPE
    if ((threads[i].spe_spu_control_area = spe_ps_area_get(threads[i].spe_context, SPE_CONTROL_AREA)) == NULL)
    {
      perror(" Failed getting spe's control state area");
    }
  }

  unsigned int mbox[1] = {0};

  getTime(0);
  while (!mbox[0])
  {
    // check if all spes arrived
    // that's for synchronization
    for (i = 0; i < numOfSPUs; i++)
    {
      while(spe_out_mbox_status(threads[i].spe_context) == 0)
        ;
      spe_out_mbox_read(threads[i].spe_context, mbox,1);
    }

    //go
    for (i = 0; i < numOfSPUs; i++)
    {
      spe_signal_write(threads[i].spe_context, SPE_SIG_NOTIFY_REG_1, 1);
    }

  }
  getTime(1);

  int res = 0;
  for (i = 0; i < numOfSPUs; i++)
  {
    if (pthread_join(threads[i].ppe_thread,NULL))
    {
      perror("Failed pthread_join");
      exit(1);
    }

    if ((res = spe_context_destroy (threads[i].spe_context)) != 0)
    {
      fprintf (stderr, "Failed spe_context_destroy(rc=%d, errno=%d, strerror=%s)\n", res, errno, strerror(errno));
      exit (1);

    }
  }


  //   plot the matrix --------------------------------------------------------------------------
#ifdef PLOT
  for ( i = 0; i < n; i++)
  {
    for ( j = 0; j < n; j++)
    {
      printf(" %2.3f ",arrayA[i][j]);
    }
    printf("\n");
  }
#endif

  return 0;

}

/*
 * =====================================================================================
 *
 *        Filename:  worker.c
 *
 *     Description: This program is part of the Gaussian elimination algorithm performed
 *                  on a SPU
 *
 *         Version:  0.9.12
 *        Revision:  none
 *        Compiler:  spu-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  ©HPCC-Space GmbH
 *
 * =====================================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spu_intrinsics.h>
#include <spu_mfcio.h>
#include <sys/types.h>
#include <malloc_align.h>
#include "cellGaussElim.h"
#include <float.h>

//#define DECTIME
//#define SYSTIME

volatile control_block cb __attribute__ ((aligned (128))) ;
volatile float *rows;
volatile float *maRow;
volatile unsigned int *rowIDs;
```

```
unsigned int gFirstDecrTime;

void initDTime()
{
  spu_writech(SPU_WrDec, -1);
  gFirstDecrTime = spu_readch(SPU_RdDec);
}

double getDTime()
{
  double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 79800000.0;
  //double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 14318000.0;
  return tRet;
}

void print_vector(char *name,vector float val)
{
  printf("Vector %s {%f, %f, %f, %f}\n",name, spu_extract(val, 0),
         spu_extract(val, 1), spu_extract(val, 2), spu_extract(val, 3));
}


int
main (unsigned long long spe_id, unsigned long long parm)
{

  unsigned int i,l;
  int tag_id = 0;
  float m;


  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) (&cb), (unsigned int) parm, sizeof (control_block),
                tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  // get the rowIDs of the rows to be copied from main memory ----------------------

  unsigned int trNumOfRows = DMA128(cb.numOfRows);
  unsigned int numOfRows = cb.numOfRows;

  rowIDs = _malloc_align(trNumOfRows * sizeof(int),7);
  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) (rowIDs), (unsigned int) cb.rowIDs, trNumOfRows * sizeof(int),
                tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  // get the rows-------------------------------------------------------------------
  unsigned int trSize = DMA128(cb.size);
  unsigned int size = cb.size;

  float *rowPointer[trSize] __attribute__ ((aligned (128)));
  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) (rowPointer), (unsigned int) (cb.arrayA),
                trSize * sizeof (float*), tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  float *rows[numOfRows];


  for(i = 0; i < numOfRows; i++)
  {
    spu_writech (MFC_WrTagMask, -1);
    rows[i] = _malloc_align(trSize * sizeof(float),7);
    spu_mfcdma32 ((void *) (rows[i]), (unsigned int) (rowPointer[rowIDs[i]]),
                  trSize * sizeof (float), tag_id, MFC_GET_CMD);
    spu_mfcstat (MFC_TAG_UPDATE_ALL);
  }


  //get master row-------------------------------------------------------------------
  unsigned int masterrow = 0;
  maRow =  (float *)_malloc_align(trSize * sizeof(float),7);
  unsigned int z = 0;
  vector float *v1;
  vector float *masterV;
  vector float v2;
  unsigned int stepsize;
  vector float mask[] = {
                          {-FLT_MAX, -FLT_MAX, -FLT_MAX, -FLT_MAX},
                          {0, -FLT_MAX, -FLT_MAX, -FLT_MAX},
                          {0, 0, -FLT_MAX, -FLT_MAX},
                          {0, 0, 0, -FLT_MAX}
                        };
#ifdef SYSTIME

  getTime(0);
#endif

#ifdef DECTIME

  initDTime();
#endif

  while (masterrow < size-1)
  {

    spu_writech (MFC_WrTagMask, -1);
    spu_mfcdma32 ((void *) (maRow), (unsigned int) (rowPointer[masterrow]),
                  trSize * sizeof(float) , tag_id, MFC_GET_CMD);
    spu_mfcstat (MFC_TAG_UPDATE_ALL);
    masterV = (vector float*) maRow;
    // compute

    for (l = 0; l < numOfRows; l++)
    {
      if (masterrow < rowIDs[l])
      {
        //m = rows[l][masterrow] / maRow[masterrow];
        stepsize = size/4 +1;
        v1 =  (vector float*)&rows[l][0];
        v2 = spu_splats((-rows[l][masterrow] / maRow[masterrow]));
        v2 = spu_and(v2, mask[masterrow % 4]);
        for ( i = (masterrow)/4; i < stepsize; i++)
        {
          //rows[l][i] =  rows[l][i] - maRow[i] * m;
          *(v1+i) = spu_madd(*(masterV+i),v2,*(v1+i));
        }
      }
      if ((masterrow+1) == rowIDs[l])
        z = l;
    }

    //write back -------------------------------------------------------------------------

    if (z)
    {
      spu_writech (MFC_WrTagMask, -1);
      spu_mfcdma32 ((void *) (rows[z]), (unsigned int) (rowPointer[rowIDs[z]]),
                    trSize * sizeof (float), tag_id, MFC_PUT_CMD);
      spu_mfcstat (MFC_TAG_UPDATE_ALL);
      z = 0;

    }
    masterrow++;


    spu_write_out_mbox(0);
    spu_read_signal1();

  }
#ifdef DECTIME

  printf("time in spu: %f\n", getDTime());
#endif


  for (i = 0; i < numOfRows; i++)
  {
    spu_writech (MFC_WrTagMask, -1);
    spu_mfcdma32 ((void *) (rows[i]), (unsigned int) (rowPointer[rowIDs[i]]),
                  trSize * sizeof (float), tag_id, MFC_PUT_CMD);
    spu_mfcstat (MFC_TAG_UPDATE_ALL);
  }

  spu_write_out_mbox(1);

#ifdef SYSTIME

  getTime(1);
#endif


  return 0;
}


//------------------------------------------------------------------------

// common includes

#include <time.h>
#include <sys/time.h>


#ifndef CELLGAUSSELIM_H_
#define CELLGAUSSELIM_H_

#define MAXROWSIZE 256

#define DMA128(n) (((int) (n / 128) +1) * 128)
#define DMA16(n) (((int) (n / 16) +1) * 16)

typedef struct _control_block
{
  float **arrayA;    //pointer to array in main memory
  int size;       // size of the array
  int *rowIDs;    // which rows to be computed
  int numOfRows;  // how many rows in total for the SPU (index for rowIDs)
}
control_block __attribute__ ((aligned (16)));
;

#define HERE printf("HERE\n");


void getTime(int print)
{
  static struct timeval t1; /* var for previous time stamp */
  static struct timeval t2; /* var of current time stamp */
  struct timezone tzp;
  if(gettimeofday(&t2, &tzp) == -1)
    exit(0);
  if(print == 1)
  {
    double elapsed_seconds=(double)(t2.tv_sec - t1.tv_sec) + ((double)(t2.tv_usec - t1.tv_usec))/1000000.0;
    printf("Time spent [%.5fs] \n", elapsed_seconds);
  }
  t1 = t2;
}

#endif /*CELLGAUSSELIM_H_*/
```

13.3. Matrix-Matrix Multiplication (1 SPE, scalar operations)

```
//------------ppu.h -------------------------------------

#include <stdio.h>
#include <libspe2.h>
#include <sched.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>
#include <pthread.h>
#include "ppuspu.h"


extern spe_program_handle_t spu;


// thread context for posix/SPE combination
typedef struct thread_data
{
  spe_context_ptr_t spe_context;
  pthread_t ppe_thread;
  unsigned int entry;
  void *argp;
}
thread_data_t;

// startup the thread on the SPE
void *ppe_pthread_function(void *arg)
{
  thread_data_t *threadData = (thread_data_t *) arg;
  if (spe_context_run(threadData->spe_context, &threadData->entry, 0,threadData->argp,NULL,NULL) < 0)
  {
    perror("Failed running context");
    exit(1);
  }

  pthread_exit(NULL);
}


//- main --------------------------------------------------------------

int
main (int argc, char **argv)
{
  int size = 40;
  thread_data_t thread;
  control_block cb;

  int arrSize = size * size;
  float arrayA[arrSize];
  float arrayB[arrSize];
  float arrayC[arrSize];

  cb.size = size;
  cb.arrayA = arrayA;
  cb.arrayB = arrayB;
  cb.arrayC = arrayC;

  int i,j;
  for (j = 0; j < size; j++)
    for (i = 0; i < size; i++)
    {
      ARR(arrayA,j,i) = 2;
      ARR(arrayB,j,i) = 2;
      ARR(arrayC,j,i) = 0;
    }
  cb.size = size;
  cb.arrayA = arrayA;
  cb.arrayB = arrayB;
  cb.arrayC = arrayC;


  // creating the SPE context
  if ((thread.spe_context = spe_context_create(SPE_MAP_PS,NULL)) == NULL)
  {
    perror("Failed creating context");
    exit(1);
  }

  // load the SPE binary into the SPE context
  if (spe_program_load(thread.spe_context, &spu))
  {
    perror("Failed loading program");
    exit(1);
  }

  // init
  thread.entry = SPE_DEFAULT_ENTRY;
  thread.argp = &cb;

  // run the thread
  if(pthread_create(&thread.ppe_thread, NULL, &ppe_pthread_function, &thread))
  {
    perror("Failed creating thread");
    exit(1);
  }


  if (pthread_join(thread.ppe_thread,NULL))
  {
    perror("Failed pthread_join");
    exit(1);
  }

  int res = 0;
  if ((res = spe_context_destroy (thread.spe_context)) != 0)
  {
    fprintf (stderr, "Failed spe_context_destroy(rc=%d, errno=%d, strerror=%s)\n", res, errno, strerror(errno));
    exit (1);

  }

//  for (j = 0; j < size; j++)
//  {
//    for (i = 0; i < size; i++)
//    {
//      printf(" %f", ARR(arrayC,j,i)) ;
//    }
//    printf("\n");
//  }


  return (0);

}




//------------spu.h ----------------------------------------------------

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spu_intrinsics.h>
#include <spu_mfcio.h>
#include <sys/types.h>
#include <malloc_align.h>
#include <ppuspu.h>



volatile control_block cb __attribute__ ((aligned (16))) ;
unsigned int gFirstDecrTime;
void initDTime()
{
  spu_writech(SPU_WrDec, -1);
  gFirstDecrTime = spu_readch(SPU_RdDec);
}

double getDTime()
{
  double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 79800000.0;
  //double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 14318000.0;
  return tRet;
}


int
main (unsigned long long spe_id, unsigned long long parm)
{
  initDTime();
  int tag_id = 0;

  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) (&cb), (unsigned int) parm, sizeof (control_block),
                tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  int size = cb.size;
  int arraySize = DMA16(size * size);

  float *arrayA = _malloc_align(arraySize * sizeof(float), 7);
  float *arrayB = _malloc_align(arraySize * sizeof(float), 7);
  float *arrayC = _malloc_align(arraySize * sizeof(float), 7);

  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) arrayA, (unsigned int) cb.arrayA, arraySize * sizeof (float),
                tag_id, MFC_GET_CMD);
  spu_mfcdma32 ((void *) arrayB, (unsigned int) cb.arrayB, arraySize * sizeof (float),
                tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);



  int i,j,k;
  int countOp = 0;

  for (j = 0; j < size; j++)
    for (i = 0; i < size; i++)
      for ( k = 0; k < size; k++)
      {
        ARR(arrayC,j,i) = ARR(arrayC,j,i) + (ARR(arrayA,j,k)) * (ARR(arrayB,k,i));

      }

  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) arrayC, (unsigned int) (cb.arrayC),
                arraySize * sizeof (float), tag_id, MFC_PUT_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  printf("time in spu: %f\n", getDTime());


  return 0;
}



//----------------------------- ppuspu.h -------------------

#ifndef PPUSPU_H_
#define PPUSPU_H_

typedef struct _control_block
{
  float *arrayA;    //pointer to array in main memory
  float *arrayB;
  float *arrayC;
  int size;       // size of the array
}
control_block __attribute__ ((aligned (16))); ;


#define ARR(uv,j,i) *((uv) + (j) * size + (i))

#define DMA128(n) (((int) (n / 128) +1) * 128)
#define DMA16(n) (((int) (n / 16) +1) * 16)


#endif /*PPUSPU_H_*/
```

13.4. Matrix-Matrix Multiplication (1 SPE, vector operations)

```
/*
 * =====================================================================================
 *
 *        Filename:  cellMaMuVec.c
 *
 *     Description:  This program performs matrix multiplication on one SPE
 *
 *         Version:  1.0.0
 *        Revision:  see svn
 *        Compiler:  ppu32-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  HPCC-Space GmbH
 *
 * =====================================================================================
 */


#include <stdio.h>
#include <libspe2.h>
#include <sched.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>
#include <pthread.h>
#include <libmisc.h>
#include <math.h>
#include "ppuspu.h"


//#define PRINTVAR

extern spe_program_handle_t spu;


// thread context for posix/SPE combination
typedef struct thread_data
{
  spe_context_ptr_t spe_context;
  pthread_t ppe_thread;
  unsigned int entry;
  void *argp;
}
thread_data_t;

// startup the thread on the SPE
void *ppe_pthread_function(void *arg)
{
  thread_data_t *threadData = (thread_data_t *) arg;
  if (spe_context_run(threadData->spe_context, &threadData->entry, 0,threadData->argp,NULL,NULL) < 0)
  {
    perror("Failed running context");
    exit(1);
  }

  pthread_exit(NULL);
}

void getTime(int print)
{
  static struct timeval t1; /* var for previous time stamp */
  static struct timeval t2; /* var of current time stamp */
  struct timezone tzp;
  if(gettimeofday(&t2, &tzp) == -1)
    exit(0);
  if(print == 1)
  {
    double elapsed_seconds=(double)(t2.tv_sec - t1.tv_sec) + ((double)(t2.tv_usec - t1.tv_usec))/1000000.0;
    printf("Time spent [%.5fs] \n", elapsed_seconds);
  }
  t1 = t2;
}

//- main --------------------------------------------------------------


int
main (int argc, char **argv)
{

  int size = 140;
  thread_data_t thread;
  control_block cb;

  int arrSize = DMA128(size * size * sizeof(float));


  float arrayA[arrSize] __attribute__ ((aligned (128)));
  float arrayB[arrSize] __attribute__ ((aligned (128)));
  float arrayC[arrSize] __attribute__ ((aligned (128)));

  cb.size = size;
  cb.arrayA = arrayA;
  cb.arrayB = arrayB;
  cb.arrayC = arrayC;


  int i,j;
  for (i = 0; i < arrSize; i++)
  {
    arrayA[i] = sqrt(2);
    arrayB[i] = sqrt(2);
    arrayC[i] = 0;
  }
  cb.size = size;
  cb.arrayA = arrayA;
  cb.arrayB = arrayB;
  cb.arrayC = arrayC;



  // creating the SPE context
  if ((thread.spe_context = spe_context_create(SPE_MAP_PS,NULL)) == NULL)
  {
    perror("Failed creating context");
    exit(1);
  }

  // load the SPE binary into the SPE context
  if (spe_program_load(thread.spe_context, &spu))
  {
    perror("Failed loading program");
    exit(1);
  }

  // init
  thread.entry = SPE_DEFAULT_ENTRY;
  thread.argp = &cb;

  // run the thread
  if(pthread_create(&thread.ppe_thread, NULL, &ppe_pthread_function, &thread))
  {
    perror("Failed creating thread");
    exit(1);
  }


  if (pthread_join(thread.ppe_thread,NULL))
  {
    perror("Failed pthread_join");
    exit(1);
  }
  //  getTime(1);
  int res = 0;
  if ((res = spe_context_destroy (thread.spe_context)) != 0)
  {
    fprintf (stderr, "Failed spe_context_destroy(rc=%d, errno=%d, strerror=%s)\n", res, errno, strerror(errno));
    exit (1);

  }
```

```
#ifdef PRINTVAR

  for (j = 0; j < size; j++)
  {
    for (i = 0; i < size; i++)
    {
      printf(" %1.1f", ARR(arrayC,j,i)) ;
    }
    printf("\n");
  }
#endif

  return (0);

}

/*
 * =====================================================================================
 *
 *        Filename:  spu.c
 *
 *     Description:  This program performs Gauss elimination
 *
 *         Version:  1.0.0
 *        Revision:  see svn
 *        Compiler:  ppu32-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  HPCC-Space GmbH
 *
 * =====================================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spu_intrinsics.h>
#include <spu_mfcio.h>
#include <sys/types.h>
#include <malloc_align.h>
#include <ppuspu.h>



volatile control_block cb __attribute__ ((aligned (16))) ;
unsigned int gFirstDecrTime;
void initDTime()
{
  spu_writech(SPU_WrDec, -1);
  gFirstDecrTime = spu_readch(SPU_RdDec);
}

double getDTime()
{
  //PS3
  double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 79800000.0;

  // Q20
  //double tRet = 1.0*(gFirstDecrTime - spu_readch(SPU_RdDec))/ 14318000.0;
  return tRet;
}



int
main (unsigned long long spe_id, unsigned long long parm)
{
  initDTime();
  unsigned int tag_id = 0;

  spu_writech (MFC_WrTagMask, -1);
  spu_mfcdma32 ((void *) (&cb), (unsigned int) parm, sizeof (control_block),
                tag_id, MFC_GET_CMD);
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  unsigned int size = cb.size;
  unsigned int vecsize = size / 4;
  unsigned int arraySize = DMA128(size *  size * sizeof(float));
  unsigned int remainChunk = arraySize;
  unsigned int getChunk = arraySize;
  unsigned int getChunkPrev = 0;

  vector float *arrayA = _malloc_align(arraySize, 7);
  vector float *arrayB = _malloc_align(arraySize, 7);
  vector float *arrayC = _malloc_align(arraySize, 7);

  do
  {
    if (remainChunk > 16384)
    {
      getChunk = 16384;
      remainChunk -= 16384;
    }

    spu_writech (MFC_WrTagMask, -1);
    spu_mfcdma32 ((void *) arrayA, (unsigned int) (cb.arrayA + getChunkPrev) , getChunk,
                  tag_id, MFC_GET_CMD);
    spu_mfcdma32 ((void *) arrayB, (unsigned int) (cb.arrayB + getChunkPrev), getChunk,
                  tag_id, MFC_GET_CMD);
    getChunkPrev += getChunk;
    getChunk = remainChunk;
  }
  while ( remainChunk > 16384 );
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  unsigned int i,j,k;

  vector float tf1 = {0.0,0.0,0.0,0.0};
  vector float tf2 = {0.0,0.0,0.0,0.0};
  vector float tf3 = {0.0,0.0,0.0,0.0};
  vector float tf4 = {0.0,0.0,0.0,0.0};
  vector float tmp1;
  vector float tmp2;
  vector float tmp3;
  vector float tmp4;

  vector float nullVect = {0.0,0.0,0.0,0.0};
  //initDTime();

  for (j = 0; j < size; j++)
  {

    for (i = 0; i < vecsize; i++)
    {
      for (k = 0; k < vecsize; k++)
      {
        tf1 = spu_madd(ARRV(arrayA,j,k) , ARRV(arrayB,i,k) , tf1 );
        tf2 = spu_madd(ARRV(arrayA,j,k) , ARRV(arrayB,i+1,k) , tf2 );
        tf3 = spu_madd(ARRV(arrayA,j,k) , ARRV(arrayB,i+2,k) , tf3 );
        tf4 = spu_madd(ARRV(arrayA,j,k) , ARRV(arrayB,i+3,k) , tf4 );
      }
      tmp1 = (vector float )
             {
               spu_extract(tf1,0), spu_extract(tf2,0) ,spu_extract(tf3,0), spu_extract(tf4,0)
             };
      tmp2 = (vector float )
             {
               spu_extract(tf1,1), spu_extract(tf2,1) ,spu_extract(tf3,1), spu_extract(tf4,1)
             };
      tmp3 = (vector float )
             {
               spu_extract(tf1,2), spu_extract(tf2,2) ,spu_extract(tf3,2), spu_extract(tf4,2)
             };
      tmp4 = (vector float )
             {
               spu_extract(tf1,3), spu_extract(tf2,3) ,spu_extract(tf3,3), spu_extract(tf4,3)
             };


      tf1 = spu_add(tmp1,tmp2);
      tf2 = spu_add(tmp3,tmp4);
      ARRV(arrayC,j, i) = spu_add(tf1  ,tf2 );


      tf1 = nullVect;
      tf2 = nullVect;
      tf3 = nullVect;
      tf4 = nullVect;

    }
  }
  //printf("time in spu: %f\n", getDTime());

  remainChunk = arraySize;
  getChunk = arraySize;
  getChunkPrev = 0;
  do
  {
    if (remainChunk > 16384)
    {
      getChunk = 16384;
      remainChunk -= 16384;
    }

    spu_writech (MFC_WrTagMask, -1);
    spu_mfcdma32 ((void *) arrayC, (unsigned int) (cb.arrayC + getChunkPrev),
                  getChunk, tag_id, MFC_PUT_CMD);

    getChunkPrev += getChunk;
    getChunk = remainChunk;

  }
  while ( remainChunk > 16384 );
  spu_mfcstat (MFC_TAG_UPDATE_ALL);

  printf("time in spu: %f\n", getDTime());


  return 0;
}

/*
 * =====================================================================================
 *
 *        Filename:  ppuspu.h
 *
 *     Description:  This file contains common elemnts
 *
 *         Version:  1.0.0
 *        Revision:  see svn
 *        Compiler:  ppu32-gcc
 *
 *          Author:  Torsten Gollnick (tg), tg@hpcc-space.de
 *         Company:  HPCC-Space GmbH
 *
 * =====================================================================================
 */


#ifndef PPUSPU_H_
#define PPUSPU_H_

typedef struct _control_block
{
  float *arrayA;    //pointer to array in main memory
  float *arrayB;
  float *arrayC;
  int size;       // size of the array
}
control_block __attribute__ ((aligned (16))); ;


#define ARR(uv,j,i) *((uv) + (j) * size + (i))
#define ARRV(uv,j,i) *((uv) + (j) * vecsize + (i))

#define DMA128(n) (((int) (n / 128) +1) * 128)
#define DMA16(n) (((int) (n / 16) +1) * 16)


#endif /*PPUSPU_H_*/
```

13.5. IBM Matrix-Matrix Multiplication

```
/* --------------------------------------------------------------  */
/* (C)Copyright 2001,2006,                                         */
/* International Business Machines Corporation,                    */
/* Sony Computer Entertainment, Incorporated,                      */
/* Toshiba Corporation,                                            */
/*                                                                 */
/* All Rights Reserved.                                            */
/* --------------------------------------------------------------  */
/* PROLOG END TAG zYx                                              */
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <fenv.h>
#include <assert.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <libspe.h>
#include <libmisc.h>
#include <math.h>
#include <string.h>
#include <errno.h>
#include <sched.h>
#include <sys/times.h>
#include <time.h>
#include <numa.h>


#define M           64        /* basic block size */
#define MAX_BLOCKS  256
#define MAX_SPUS    16
#define ERR_THRESHOLD         0.01

extern spe_program_handle_t block;
extern int daxpy(double *, double *, double *, int, int, double, double, double);

/* Per thread state
*/
struct _threads {
  speid_t id;                                              // spe thread id;
  spe_spu_control_area_t *ctl_area;              // pointer to control ps area
  int in_cnt;                                              // inbound mailbox available element count
} threads[MAX_SPUS];


float *tmp = NULL;

int use_heap = 0;
char *mem_file = "/huge/daxpy_mem.bin";
char *mem_addr = NULL;

void print_usage(char * name)
{
  printf("USAGE: %s [options]\n", name);
  printf("       Valid Options include:\n");
  printf("         -i #   : perform specified number of iterations. Default is 1.\n");
  printf("         -h     : output this usage help screen.\n");
  printf("         -H     : heap allocate all buffers. Default is to use the huge tlbfs.\n");
  printf("         -n     : use local numa binding to maximize performance by duplicating\n");
  printf("                  the input arrays in each node's memory and binding half the\n");
  printf("                  SPU threads on each node.\n");
  printf("                  SPUs is greater than 8.\n");
  printf("         -m #   : perform maxtrix multiply of matrices of size #. Valid sizes\n");
  printf("                  are 64, 128, 256, 512, 1024, 2048, and 4096. Default is 64.\n");
  printf("         -p     : display performance statistics\n");
  printf("         -s #   : run with # number of SPUs. The dafault is 1 SPU.\n");
  printf("         -v     : verify the final matrix multiply result.\n");
  printf("         -V     : verbose output.\n");
  exit(1);
}
```

```
void send_mail(int idx, unsigned int data)
{
  volatile spe_spu_control_area_t *ctl;

  ctl = threads[idx].ctl_area;

  if (threads[idx].in_cnt == 0) {
    /* Wait for available space in the inbound mailbox
    */
    while ((threads[idx].in_cnt = (ctl->SPU_Mbox_Stat >> 8) & 0xFF) == 0);
  }

  /* Place the data into the thread inbound mailbox
  */
  ctl->SPU_In_Mbox = data;
  threads[idx].in_cnt--;
}


void block_swizzle(float *mat, int size)
{
  int i, j, k, l;
  float *ptr, *src, *dst;

  ptr = tmp;
  for(i=0; i<(size/M); i++) {
    for(j=0; j<(size/M); j++){
      for(k=0; k<M; k++) {
        for(l=0; l<M; l++) {
          *ptr++ = mat[i*(M*size)+j*M+k*size+l];
        }
      }
    }
  }

  src = tmp;
  dst = mat;
  for(i=0; i<size*size; i++) *dst++ = *src++;
}


#define HUGE_PAGE_SIZE        16*1024*1024

/* Allocate a cacheline aligned memory buffer either from large pages or the
* malloc heap on the specified node.
*/
char * allocate_buffer(int size, nodemask_t *mask)
{
  char *addr;
  int fmem = -1;
  int huge_size;

  if (mask) {
    numa_set_membind(mask);
  }

  if (!use_heap) {
    if ((fmem = open (mem_file, O_CREAT | O_RDWR, 0755)) == -1) {
      printf("WARNING: unable to open file %s (errno=%d). Using malloc heap.\n", mem_file, errno);
    }
  }

  if (fmem == -1) {
    addr = (char *)malloc_align(size, 7);
    if (addr == NULL) {
      printf("ERROR: unable to malloc %d byte buffer.\n", size);
      exit(1);
    }
  } else {
    /* Delete file so that huge pages will get freed on program
    * termination.
    */
    remove(mem_file);
    huge_size = (size + HUGE_PAGE_SIZE-1) & ~(HUGE_PAGE_SIZE-1);

    assert(HUGE_PAGE_SIZE > 32*1024);
    addr = (char *) mmap (0, huge_size, PROT_READ | PROT_WRITE, MAP_PRIVATE, fmem, 0);
    if (mem_addr == MAP_FAILED) {
      printf("ERROR: unable to mmap file %s (errno=%d).\n", mem_file, errno);
      close (fmem);
      exit (1);
    }
  }
  /* Perform a memset to ensure the memory binding.
  */
  if (mask) {
    (void *)memset(addr, 0, size);
  }

  return addr;
}

int main(int argc, char *argv[])
{
  int i, j, k;
  int verify = 0;
  int verbose = 0;
  int fail = 0;
  int msize = M, blocks;
  int spus = 1;
  int iterations = 1;
  int use_numa = 0;
  int performance = 0;
  int nodes;
  spe_gid_t spe_gid;
  float *a, *b, *c, *exp;
  float *a2, *b2;
  int *cntr;
  unsigned int offset, offset2;
  int status;
  struct tms tbuf;
  int gtime;
  nodemask_t mask0, mask1;
  double elapsed_time;

  for (i=1; i<argc; i++) {
    if (*argv[i] == '-') {
      switch (*(argv[i]+1)) {
        case 'i':
          i++;
          if (i < argc) {
            iterations = atoi(argv[i]);
            if (iterations < 0) iterations = 0;
          } else {
            printf("ERROR: Number of iterations is not specified.\n");
            print_usage(argv[0]);
          }
          break;
          case 'm':           /* Run matrix of specified size */
            i++;
            if (i < argc) {
              msize = atoi(argv[i]);
              if ((msize < 64) || (msize > 4096) || (msize & (msize - 1))) {
                printf("ERROR: Invalid matrix size of %d specified\n", msize);
                print_usage(argv[0]);
              }
            } else {
              printf("ERROR: Matrix size is not specifed\n");
              print_usage(argv[0]);
            }
            break;
            case 's':                   /* Run on specified number of SPUs */
              i++;
              if (i < argc) {
                int phys_spus;

                phys_spus = spe_count_physical_spes();

                spus = atoi(argv[i]);
                if (spus > MAX_SPUS) spus = MAX_SPUS;
                if (spus > phys_spus) spus = phys_spus;

              } else {
                printf("ERROR: Number of SPUs to use is not specified.\n");
                print_usage(argv[0]);
              }
              break;
        case 'n':
          if (numa_available() >= 0) {
            nodes = numa_max_node() + 1;
            if (nodes > 1) {
              use_numa = 1;
              nodemask_zero(&mask0);
              nodemask_set(&mask0, 0);
              nodemask_zero(&mask1);
              nodemask_set(&mask1, 1);
            } else {
              printf("WARNING: insufficient numa nodes (nodes = %d). Ignoring request.\n", nodes);
            }
          } else {
            printf("WARNING: numa is not available. Ignoring request.\n");
          }
          break;
          case 'v':           /* Let the PPU verify the final product result. */
            verify = 1;
            break;
            case 'V':                   /* Verbose output */
              verbose = 1;
              break;
              case 'H':                 /* Allocate memory buffers from the malloc heap */
                use_heap = 1;
                break;
                case 'p':               /* Display performance statistics */
                  performance = 1;
                  break;
        case 'h':
        default:
          print_usage(argv[0]);
          break;
      }
    } else {
      print_usage(argv[0]);
    }
  }

  blocks = (msize/M) * (msize/M);

  spe_gid = spe_create_group(SCHED_OTHER, 0, 1);

  /* Allocate memory buffers.
  */
  offset = 0;

  a = a2 = (float *)offset;
  offset += sizeof(float) * msize * msize;
```

```
  b = b2 = (float *)(offset);
  offset += sizeof(float) * msize * msize;
  offset2 = offset;
  c = (float *)(offset);
  offset += sizeof(float) * msize * msize;
  cntr = (int *)(offset);
  offset += 128;

  /* Create a large contiguous memory buffer by allocating a large
  * page (or more). Large page memory will also reduce the TLB thrashing.
  */
  mem_addr = allocate_buffer(offset, (use_numa) ? &mask0 : NULL);

  /* Correct the buffer pointers
  */
  a = (float  *)(mem_addr + (unsigned int)a);
  b = (float  *)(mem_addr + (unsigned int)b);
  c = (float  *)(mem_addr + (unsigned int)c);
  cntr = (int *)(mem_addr + (unsigned int)cntr);

  /* Construct matrices and expected results
  */
  exp = (float *)malloc(sizeof(float) * msize * msize);
  tmp = (float *)malloc(sizeof(float) * msize * msize);
  if ((exp == NULL) || (tmp == NULL)) {
    printf("ERROR: failed to allocated verification buffers exp and tmp\n");
    exit(1);
  }

  if (use_numa) {
    mem_addr = allocate_buffer(offset2, &mask1);

    a2 = (float  *)(mem_addr + (unsigned int)a2);
    b2 = (float  *)(mem_addr + (unsigned int)b2);
  }



/* Initialize the matrix data and counter.
  */
  printf("Initializing Arrays ... "); fflush(stdout);
  for(i=0; i<msize; i++) {
    for(j=0; j<msize; j++) {
      a[i*msize+j] = rand_0_to_1();
      b[i*msize+j] = rand_0_to_1();
      c[i*msize+j] = 0.0f;

      if (use_numa) {
        a2[i*msize+j] = a[i*msize+j];
        b2[i*msize+j] = b[i*msize+j];
      }
    }
  }
  printf("done\n"); fflush(stdout);
  cntr[0] = 0;      /* initialize the counters */
  cntr[4] = 0;

  if (verify) {
    /* Compute expected result.
    */
    printf("Computing expected results ... "); fflush(stdout);
    for (i=0; i<msize; i++) {
      for (j=0; j<msize; j++) {
        exp[i*msize+j] = 0.0f;
        for (k=0; k<msize; k++) {
          exp[i*msize+j] += a[i*msize+k] * b[k*msize+j];
        }
      }
    }
    /* Swizzle inputs matrices and expected results to correspond to tiling.
    */
    block_swizzle(a, msize);
    block_swizzle(b, msize);
    block_swizzle(exp, msize);

    if (use_numa) {
      block_swizzle(a2, msize);
      block_swizzle(b2, msize);
    }

    printf("done\n"); fflush(stdout);
  }

  printf("Running test ... "); fflush(stdout);
  /* Start all the SPUs computing the matrix product.
  */
  /* Create each of the SPU threads
  */
  for (i=0; i<spus; i++) {
    if (use_numa) {
      if (i < spus/2) {
        numa_bind(&mask0);
      } else {
        numa_bind(&mask1);
      }
    }
    if ((threads[i].id = spe_create_thread(spe_gid, &block, 0, 0, -1, SPE_MAP_PS)) == NULL) {
      printf("INTERNAL ERROR: failed to create spu thread %d. Error = %s\n", i, strerror(errno));
      exit(1);
    }
    if ((threads[i].ctl_area = (spe_spu_control_area_t *)spe_get_ps_area(threads[i].id, SPE_CONTROL_AREA)) == NULL) {
      printf("INTERNAL ERROR: failed to get control problem state area for thread %d. Error = %s\n", i, strerror(errno));
      exit(1);
    }
    threads[i].in_cnt = 0;
  }

  *cntr = 0;

  /* Start time
  */
  gtime = times(&tbuf);

  /* Send each of the SPUs the input parameters
  */
  for (i=0; i<spus; i++) {
    send_mail(i, (unsigned int)msize);
    send_mail(i, (unsigned int)iterations);
    send_mail(i, (unsigned int)cntr);
    if ((use_numa) && (i>=(spus+1)/2)) {
      send_mail(i, (unsigned int)a2);
      send_mail(i, (unsigned int)b2);
    } else {
      send_mail(i, (unsigned int)a);
      send_mail(i, (unsigned int)b);
    }
    send_mail(i, (unsigned int)c);
  }

  /* Wait for the SPUs to complete
  */
  for (i=0; i<spus; i++) {
    (void)spe_wait(threads[i].id, &status, 0);
  }

  /* Stop time
  */
  elapsed_time = (double)(times(&tbuf) - gtime) / (double)sysconf(_SC_CLK_TCK);

  printf("done\n");


  /* Output SPU, matrix multiply results
  */
  if (verify) {
    float delta;

    printf("Verifying the results ... ");

    /* Verify the resulting matrix output.
    */
    for (i=0; i<msize; i++) {
      for (j=0; j<msize; j++) {
        delta = exp[i*msize+j] - c[i*msize+j];
        if (delta < 0.0f) delta = -delta;
        if (delta > ERR_THRESHOLD) {
          if (verbose) printf("  %d %d exp=%f got=%f\n", i, j, exp[i*msize+j], c[i*msize+j]);
          fail++;
        }
      }
    }

    if (fail) {
      printf("FAILED (%d)\n", fail);
      fail = 1;
    } else {
      printf("PASSED\n");
    }
  }

  /* Display performance statistics if requested.
  */
  if (performance) {
    printf("Performance Statistics:\n");
    printf("  number of SPEs     = %d\n", spus);
    printf("  execution time     = %.2f seconds\n", elapsed_time);
      printf("  computation rate     = %.2f GFlops/sec\n", ((double)(2*msize-1) * (double)(msize*msize) * (double)iterations) /
(1000000000.0 * elapsed_time));
      printf("  data transfer rate = %.2f GBytes/sec\n", ((double)(2*msize/M + 1) * (double)(M*M*sizeof(float)) * blocks *
(double)iterations) / (1000000000.0 * elapsed_time));
  }

  return (fail);
}


/* --------------------------------------------------------------  */
/* (C)Copyright 2001,2006,                                         */
/* International Business Machines Corporation,                    */
/* Sony Computer Entertainment, Incorporated,                      */
/* Toshiba Corporation,                                            */
/*                                                                 */
/* All Rights Reserved.                                            */
/* --------------------------------------------------------------  */
/* PROLOG END TAG zYx                                              */

/*
 * Matrix Multiply --- EUC
 *
 * block.c - Matrix Multiply with block partitioning
 */

#include <spu_intrinsics.h>
#include <stdlib.h>
#include <vec_literal.h>
#include <spu_mfcio.h>
```

```
#include "atomic_inc_return.h"
#include <stdio.h>

#ifndef M
#define M                     64                /* Size of the matrix block - M x M */
#endif

#define MAX_N                 1024
#define MAX_TILES   (MAX_N / M)

static unsigned int N;
static unsigned int ITER;
static unsigned int A_AREA, B_AREA, C_AREA;
static unsigned int FINC_AREA;

#define DIN_TAG   0
#define DOUT_TAG  2

#define DMA_Wait(TAG_ID)                          \
{                                                           \
  mfc_write_tag_mask((1<<(TAG_ID)));              \
  mfc_read_tag_status_all();                      \
}

#define SwapInBuf()                               \
{                                                           \
  OpA = InTag ? blkA1 : blkA0;                              \
  OpB = InTag ? blkB1 : blkB0;                              \
  InTag ^= 1;                                               \
  InA = InTag ? blkA1 : blkA0;                              \
  InB = InTag ? blkB1 : blkB0;                              \
}

#define SwapOutBuf()                                        \
{                                                           \
  OpC = (OutTag==DOUT_TAG) ? blkC1 : blkC0;       \
  OutC = (OutTag==DOUT_TAG) ? blkC1 : blkC0;      \
  OutTag ^= 1;                                              \
}

#ifdef USE_INLINE_ASM

#define ALIGN8B     asm volatile(".align 3")

#define SPU_FMA(RT,RA,RB,RC)                                                   \
  asm volatile(".align 3");                                                    \
  asm volatile("fma %0,%1,%2,%3":"=r"(RT):"r"(RA),"r"(RB),"r"(RC))

#define SPU_FM(RT,RA,RB)                                    \
  asm volatile(".align 3");                                 \
  asm volatile("fm %0,%1,%2":"=r"(RT):"r"(RA),"r"(RB))

#define SPU_LNOP    asm volatile("lnop")

#else

#define ALIGN8B
#define SPU_FMA(RT,RA,RB,RC)  RT = spu_madd(RA, RB, RC)
#define SPU_FM(RT,RA,RB)      RT = spu_mul(RA, RB)
#define SPU_LNOP

#endif



#define StageCBAclr(OFFSET)                                                                                  \
{                                                                                                                      \
  ALIGN8B;                                                                                                             \
  SPU_FM(c0_0B,a00,b0_0B);                                                                                   \
  SPU_FM(c1_0B,a10,b0_0B);                                                                                   \
  SPU_FM(c2_0B,a20,b0_0B);                                                                                   \
  SPU_FM(c3_0B,a30,b0_0B);                                                                                   \
  SPU_FM(c0_1B,a00,b0_1B);                                                                                   \
  SPU_FM(c1_1B,a10,b0_1B);                                                                                   \
  SPU_FM(c2_1B,a20,b0_1B);                                                                                   \
  SPU_FM(c3_1B,a30,b0_1B);                                                                                   \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B);                                                                            \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B);                                                                            \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); b0_0C = *((volatile vector float *)(ptrB+OFFSET+16));            \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); SPU_LNOP;                                                                  \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); b1_0C = *((volatile vector float *)(ptrB+M+OFFSET+16));          \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); b2_0C = *((volatile vector float *)(ptrB+2*M+OFFSET+16));        \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); b3_0C = *((volatile vector float *)(ptrB+3*M+OFFSET+16));        \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); SPU_LNOP;                                                                  \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); b0_1C = *((volatile vector float *)(ptrB+OFFSET+20));            \
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); b1_1C = *((volatile vector float *)(ptrB+M+OFFSET+20));          \
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); b2_1C = *((volatile vector float *)(ptrB+2*M+OFFSET+20));        \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); SPU_LNOP;                                                                  \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); b3_1C = *((volatile vector float *)(ptrB+3*M+OFFSET+20));        \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); *((volatile vector float *)(ptrC+OFFSET)) = c0_0A;               \
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0A;             \
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); SPU_LNOP;                                                                  \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0A; \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0A; \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1A;             \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); SPU_LNOP;                                                                  \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1A; \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1A;         \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1A;         \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); SPU_LNOP;                                                                  \
}

#define StageACBclr(OFFSET)                                                                                  \
{                                                                                                                       \
  SPU_FM(c0_0C,a00,b0_0C);                                                                                    \
  SPU_FM(c1_0C,a10,b0_0C);                                                                                    \
  SPU_FM(c2_0C,a20,b0_0C);                                                                                    \
  SPU_FM(c3_0C,a30,b0_0C);                                                                                    \
  SPU_FM(c0_1C,a00,b0_1C);                                                                                    \
  SPU_FM(c1_1C,a10,b0_1C);                                                                                    \
  SPU_FM(c2_1C,a20,b0_1C);                                                                                    \
  SPU_FM(c3_1C,a30,b0_1C);                                                                                    \
  SPU_FMA(c0_0C,a01,b1_0C,c0_0C);                                                                             \
  SPU_FMA(c1_0C,a11,b1_0C,c1_0C);                                                                             \
  SPU_FMA(c2_0C,a21,b1_0C,c2_0C); b0_0A = *((volatile vector float *)(ptrB+OFFSET+16));             \
  SPU_FMA(c3_0C,a31,b1_0C,c3_0C); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1C,a01,b1_1C,c0_1C); b1_0A = *((volatile vector float *)(ptrB+M+OFFSET+16));           \
  SPU_FMA(c1_1C,a11,b1_1C,c1_1C); b2_0A = *((volatile vector float *)(ptrB+2*M+OFFSET+16));         \
  SPU_FMA(c2_1C,a21,b1_1C,c2_1C); b3_0A = *((volatile vector float *)(ptrB+3*M+OFFSET+16));         \
  SPU_FMA(c3_1C,a31,b1_1C,c3_1C); SPU_LNOP;                                                                   \
  SPU_FMA(c0_0C,a02,b2_0C,c0_0C); b0_1A = *((volatile vector float *)(ptrB+OFFSET+20));             \
  SPU_FMA(c1_0C,a12,b2_0C,c1_0C); b1_1A = *((volatile vector float *)(ptrB+M+OFFSET+20));           \
  SPU_FMA(c2_0C,a22,b2_0C,c2_0C); b2_1A = *((volatile vector float *)(ptrB+2*M+OFFSET+20));         \
  SPU_FMA(c3_0C,a32,b2_0C,c3_0C); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1C,a02,b2_1C,c0_1C); b3_1A = *((volatile vector float *)(ptrB+3*M+OFFSET+20));         \
  SPU_FMA(c1_1C,a12,b2_1C,c1_1C); *((volatile vector float *)(ptrC+OFFSET)) = c0_0B;                \
  SPU_FMA(c2_1C,a22,b2_1C,c2_1C); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0B;              \
  SPU_FMA(c3_1C,a32,b2_1C,c3_1C); SPU_LNOP;                                                                   \
  SPU_FMA(c0_0C,a03,b3_0C,c0_0C); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0B; \
  SPU_FMA(c1_0C,a13,b3_0C,c1_0C); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0B; \
  SPU_FMA(c2_0C,a23,b3_0C,c2_0C); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1B;              \
  SPU_FMA(c3_0C,a33,b3_0C,c3_0C); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1C,a03,b3_1C,c0_1C); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1B; \
  SPU_FMA(c1_1C,a13,b3_1C,c1_1C); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1B;          \
  SPU_FMA(c2_1C,a23,b3_1C,c2_1C); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1B;          \
  SPU_FMA(c3_1C,a33,b3_1C,c3_1C); SPU_LNOP;                                                                   \
}

#define StageBACclr(OFFSET)                                                                                   \
{                                                                                                                       \
  SPU_FM(c0_0A,a00,b0_0A);                                                                                    \
  SPU_FM(c1_0A,a10,b0_0A);                                                                                    \
  SPU_FM(c2_0A,a20,b0_0A);                                                                                    \
  SPU_FM(c3_0A,a30,b0_0A);                                                                                    \
  SPU_FM(c0_1A,a00,b0_1A);                                                                                    \
  SPU_FM(c1_1A,a10,b0_1A);                                                                                    \
  SPU_FM(c2_1A,a20,b0_1A);                                                                                    \
  SPU_FM(c3_1A,a30,b0_1A);                                                                                    \
  SPU_FMA(c0_0A,a01,b1_0A,c0_0A);                                                                             \
  SPU_FMA(c1_0A,a11,b1_0A,c1_0A);                                                                             \
  SPU_FMA(c2_0A,a21,b1_0A,c2_0A); b0_0B = *((volatile vector float *)(ptrB+OFFSET+16));             \
  SPU_FMA(c3_0A,a31,b1_0A,c3_0A); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1A,a01,b1_1A,c0_1A); b1_0B = *((volatile vector float *)(ptrB+M+OFFSET+16));           \
  SPU_FMA(c1_1A,a11,b1_1A,c1_1A); b2_0B = *((volatile vector float *)(ptrB+2*M+OFFSET+16));         \
  SPU_FMA(c2_1A,a21,b1_1A,c2_1A); b3_0B = *((volatile vector float *)(ptrB+3*M+OFFSET+16));         \
  SPU_FMA(c3_1A,a31,b1_1A,c3_1A); SPU_LNOP;                                                                   \
  SPU_FMA(c0_0A,a02,b2_0A,c0_0A); b0_1B = *((volatile vector float *)(ptrB+OFFSET+20));             \
  SPU_FMA(c1_0A,a12,b2_0A,c1_0A); b1_1B = *((volatile vector float *)(ptrB+M+OFFSET+20));           \
  SPU_FMA(c2_0A,a22,b2_0A,c2_0A); b2_1B = *((volatile vector float *)(ptrB+2*M+OFFSET+20));         \
  SPU_FMA(c3_0A,a32,b2_0A,c3_0A); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1A,a02,b2_1A,c0_1A); b3_1B = *((volatile vector float *)(ptrB+3*M+OFFSET+20));         \
  SPU_FMA(c1_1A,a12,b2_1A,c1_1A); *((volatile vector float *)(ptrC+OFFSET)) = c0_0C;                \
  SPU_FMA(c2_1A,a22,b2_1A,c2_1A); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0C;              \
  SPU_FMA(c3_1A,a32,b2_1A,c3_1A); SPU_LNOP;                                                                   \
  SPU_FMA(c0_0A,a03,b3_0A,c0_0A); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0C; \
  SPU_FMA(c1_0A,a13,b3_0A,c1_0A); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0C; \
  SPU_FMA(c2_0A,a23,b3_0A,c2_0A); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1C;              \
  SPU_FMA(c3_0A,a33,b3_0A,c3_0A); SPU_LNOP;                                                                   \
  SPU_FMA(c0_1A,a03,b3_1A,c0_1A); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1C; \
  SPU_FMA(c1_1A,a13,b3_1A,c1_1A); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1C;          \
  SPU_FMA(c2_1A,a23,b3_1A,c2_1A); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1C;          \
  SPU_FMA(c3_1A,a33,b3_1A,c3_1A); SPU_LNOP;                                                                   \
}

#define StageCBA(OFFSET,OFFB)                                                                                           \
{                                                                                                                                 \
  ALIGN8B;                                                                                                                        \
  SPU_FMA(c0_0B,a00,b0_0B,c0_0B); c0_0C = *((volatile vector float *)(ptrC+OFFSET+16));                       \
  SPU_FMA(c1_0B,a10,b0_0B,c1_0B); c1_0C = *((volatile vector float *)(ptrC+M+OFFSET+16));                     \
  SPU_FMA(c2_0B,a20,b0_0B,c2_0B); c2_0C = *((volatile vector float *)(ptrC+2*M+OFFSET+16));                   \
  SPU_FMA(c3_0B,a30,b0_0B,c3_0B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_1B,a00,b0_1B,c0_1B); c3_0C = *((volatile vector float *)(ptrC+3*M+OFFSET+16));                   \
  SPU_FMA(c1_1B,a10,b0_1B,c1_1B); c0_1C = *((volatile vector float *)(ptrC+OFFSET+20));                       \
  SPU_FMA(c2_1B,a20,b0_1B,c2_1B); c1_1C = *((volatile vector float *)(ptrC+M+OFFSET+20));                     \
  SPU_FMA(c3_1B,a30,b0_1B,c3_1B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B); c2_1C = *((volatile vector float *)(ptrC+2*M+OFFSET+20));                   \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B); c3_1C = *((volatile vector float *)(ptrC+3*M+OFFSET+20));                   \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); b0_0C = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+16));                \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); b1_0C = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+16));    \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); b2_0C = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+16)); \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); b3_0C = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+16)); \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); b0_1C = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+20));                \
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); b1_1C = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+20));    \
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); b2_1C = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+20)); \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); b3_1C = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+20)); \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); *((volatile vector float *)(ptrC+OFFSET)) = c0_0A;                          \
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0A;                        \
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0A;            \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0A;            \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1A;                        \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); SPU_LNOP;                                                                             \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1A;            \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1A;                  \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1A;                  \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); SPU_LNOP;                                                                           \
}

#define StageCBAmod(OFFSET,OFFB)                                                                                      \
{                                                                                                                               \
  ALIGN8B;                                                                                                                      \
  SPU_FMA(c0_0B,a00,b0_0B,c0_0B); SPU_LNOP;                                                                           \
  SPU_FMA(c1_0B,a10,b0_0B,c1_0B); b2_0B = *((volatile vector float *)(ptrB+2*M+OFFB*M+8));                  \
  SPU_FMA(c2_0B,a20,b0_0B,c2_0B); b2_1B = *((volatile vector float *)(ptrB+2*M+OFFB*M+12));                 \
  SPU_FMA(c3_0B,a30,b0_0B,c3_0B); b3_0B = *((volatile vector float *)(ptrB+3*M+OFFB*M+8));                  \
  SPU_FMA(c0_1B,a00,b0_1B,c0_1B); b3_1B = *((volatile vector float *)(ptrB+3*M+OFFB*M+12));                 \
  SPU_FMA(c1_1B,a10,b0_1B,c1_1B); c0_0C = *((volatile vector float *)(ptrC+OFFSET+16));                     \
  SPU_FMA(c2_1B,a20,b0_1B,c2_1B); c1_0C = *((volatile vector float *)(ptrC+M+OFFSET+16));                   \
  SPU_FMA(c3_1B,a30,b0_1B,c3_1B); c2_0C = *((volatile vector float *)(ptrC+2*M+OFFSET+16));                 \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B); c3_0C = *((volatile vector float *)(ptrC+3*M+OFFSET+16));                 \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B); SPU_LNOP;                                                                           \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); c0_1C = *((volatile vector float *)(ptrC+OFFSET+20));                     \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); c1_1C = *((volatile vector float *)(ptrC+M+OFFSET+20));                   \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); c2_1C = *((volatile vector float *)(ptrC+2*M+OFFSET+20));                 \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); c3_1C = *((volatile vector float *)(ptrC+3*M+OFFSET+20));                 \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); b0_0C = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+16));              \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); b1_0C = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+16));  \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); b2_0C = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); b3_0C = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); SPU_LNOP;                                                                           \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); b0_1C = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+20));              \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); b1_1C = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+20));  \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); b2_1C = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); b3_1C = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); *((volatile vector float *)(ptrC+OFFSET)) = c0_0A;                        \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0A;                      \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0A;          \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0A;          \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1A;                      \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1A;          \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1A;                  \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1A;                  \
}

#define StageACB(OFFSET,OFFB)                                                                                         \
{                                                                                                                               \
  SPU_FMA(c0_0C,a00,b0_0C,c0_0C); c0_0A = *((volatile vector float *)(ptrC+OFFSET+16));                     \
  SPU_FMA(c1_0C,a10,b0_0C,c1_0C); c1_0A = *((volatile vector float *)(ptrC+M+OFFSET+16));                   \
  SPU_FMA(c2_0C,a20,b0_0C,c2_0C); c2_0A = *((volatile vector float *)(ptrC+2*M+OFFSET+16));                 \
  SPU_FMA(c3_0C,a30,b0_0C,c3_0C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1C,a00,b0_1C,c0_1C); c3_0A = *((volatile vector float *)(ptrC+3*M+OFFSET+16));                 \
  SPU_FMA(c1_1C,a10,b0_1C,c1_1C); c0_1A = *((volatile vector float *)(ptrC+OFFSET+20));                     \
  SPU_FMA(c2_1C,a20,b0_1C,c2_1C); c1_1A = *((volatile vector float *)(ptrC+M+OFFSET+20));                   \
  SPU_FMA(c3_1C,a30,b0_1C,c3_1C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_0C,a01,b1_0C,c0_0C); c2_1A = *((volatile vector float *)(ptrC+2*M+OFFSET+20));                 \
  SPU_FMA(c1_0C,a11,b1_0C,c1_0C); c3_1A = *((volatile vector float *)(ptrC+3*M+OFFSET+20));                 \
  SPU_FMA(c2_0C,a21,b1_0C,c2_0C); b0_0A = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+16));              \
  SPU_FMA(c3_0C,a31,b1_0C,c3_0C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1C,a01,b1_1C,c0_1C); b1_0A = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+16));  \
  SPU_FMA(c1_1C,a11,b1_1C,c1_1C); b2_0A = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c2_1C,a21,b1_1C,c2_1C); b3_0A = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c3_1C,a31,b1_1C,c3_1C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_0C,a02,b2_0C,c0_0C); b0_1A = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+20));              \
  SPU_FMA(c1_0C,a12,b2_0C,c1_0C); b1_1A = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+20));  \
  SPU_FMA(c2_0C,a22,b2_0C,c2_0C); b2_1A = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c3_0C,a32,b2_0C,c3_0C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1C,a02,b2_1C,c0_1C); b3_1A = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c1_1C,a12,b2_1C,c1_1C); *((volatile vector float *)(ptrC+OFFSET)) = c0_0B;                        \
  SPU_FMA(c2_1C,a22,b2_1C,c2_1C); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0B;                      \
  SPU_FMA(c3_1C,a32,b2_1C,c3_1C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_0C,a03,b3_0C,c0_0C); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0B;          \
  SPU_FMA(c1_0C,a13,b3_0C,c1_0C); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0B;          \
  SPU_FMA(c2_0C,a23,b3_0C,c2_0C); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1B;                      \
  SPU_FMA(c3_0C,a33,b3_0C,c3_0C); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1C,a03,b3_1C,c0_1C); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1B;          \
  SPU_FMA(c1_1C,a13,b3_1C,c1_1C); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1B;                  \
  SPU_FMA(c2_1C,a23,b3_1C,c2_1C); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1B;                  \
  SPU_FMA(c3_1C,a33,b3_1C,c3_1C); SPU_LNOP;                                                                           \
}

#define StageBAC(OFFSET,OFFB)                                                                                         \
{                                                                                                                               \
  SPU_FMA(c0_0A,a00,b0_0A,c0_0A); c0_0B = *((volatile vector float *)(ptrC+OFFSET+16));                     \
  SPU_FMA(c1_0A,a10,b0_0A,c1_0A); c1_0B = *((volatile vector float *)(ptrC+M+OFFSET+16));                   \
  SPU_FMA(c2_0A,a20,b0_0A,c2_0A); c2_0B = *((volatile vector float *)(ptrC+2*M+OFFSET+16));                 \
  SPU_FMA(c3_0A,a30,b0_0A,c3_0A); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1A,a00,b0_1A,c0_1A); c3_0B = *((volatile vector float *)(ptrC+3*M+OFFSET+16));                 \
  SPU_FMA(c1_1A,a10,b0_1A,c1_1A); c0_1B = *((volatile vector float *)(ptrC+OFFSET+20));                     \
  SPU_FMA(c2_1A,a20,b0_1A,c2_1A); c1_1B = *((volatile vector float *)(ptrC+M+OFFSET+20));                   \
  SPU_FMA(c3_1A,a30,b0_1A,c3_1A); SPU_LNOP;                                                                           \
  SPU_FMA(c0_0A,a01,b1_0A,c0_0A); c2_1B = *((volatile vector float *)(ptrC+2*M+OFFSET+20));                 \
  SPU_FMA(c1_0A,a11,b1_0A,c1_0A); c3_1B = *((volatile vector float *)(ptrC+3*M+OFFSET+20));                 \
  SPU_FMA(c2_0A,a21,b1_0A,c2_0A); b0_0B = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+16));              \
  SPU_FMA(c3_0A,a31,b1_0A,c3_0A); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1A,a01,b1_1A,c0_1A); b1_0B = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+16));  \
  SPU_FMA(c1_1A,a11,b1_1A,c1_1A); b2_0B = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c2_1A,a21,b1_1A,c2_1A); b3_0B = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+16));\
  SPU_FMA(c3_1A,a31,b1_1A,c3_1A); SPU_LNOP;                                                                           \
  SPU_FMA(c0_0A,a02,b2_0A,c0_0A); b0_1B = *((volatile vector float *)(ptrB+OFFSET+OFFB*M+20));              \
  SPU_FMA(c1_0A,a12,b2_0A,c1_0A); b1_1B = *((volatile vector float *)(ptrB+M+OFFSET+OFFB*M+20));  \
  SPU_FMA(c2_0A,a22,b2_0A,c2_0A); b2_1B = *((volatile vector float *)(ptrB+2*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c3_0A,a32,b2_0A,c3_0A); SPU_LNOP;                                                                           \
  SPU_FMA(c0_1A,a02,b2_1A,c0_1A); b3_1B = *((volatile vector float *)(ptrB+3*M+OFFSET+OFFB*M+20));\
  SPU_FMA(c1_1A,a12,b2_1A,c1_1A); *((volatile vector float *)(ptrC+OFFSET)) = c0_0C;                        \
  SPU_FMA(c2_1A,a22,b2_1A,c2_1A); *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0C;                      \
  SPU_FMA(c3_1A,a32,b2_1A,c3_1A); SPU_LNOP;                                                            \
  SPU_FMA(c0_0A,a03,b3_0A,c0_0A); *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0C;          \
  SPU_FMA(c1_0A,a13,b3_0A,c1_0A); *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0C;          \
  SPU_FMA(c2_0A,a23,b3_0A,c2_0A); *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1C;                      \
  SPU_FMA(c3_0A,a33,b3_0A,c3_0A); SPU_LNOP;                                                            \
  SPU_FMA(c0_1A,a03,b3_1A,c0_1A); *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1C;          \
  SPU_FMA(c1_1A,a13,b3_1A,c1_1A); *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1C;                  \
  SPU_FMA(c2_1A,a23,b3_1A,c2_1A); *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1C;                  \
  SPU_FMA(c3_1A,a33,b3_1A,c3_1A); SPU_LNOP;                                                            \
}

#define StageMISC(OFFA,OFFB)                                                                                 \
{                                                                                                                      \
  SPU_FMA(c0_0B,a00,b0_0B,c0_0B); a0 = *((volatile vector float *)(ptrA+OFFA+4));                  \
  SPU_FMA(c1_0B,a10,b0_0B,c1_0B); a1 = *((volatile vector float *)(ptrA+M+OFFA+4));                \
  SPU_FMA(c2_0B,a20,b0_0B,c2_0B); a2 = *((volatile vector float *)(ptrA+2*M+OFFA+4));              \
  SPU_FMA(c3_0B,a30,b0_0B,c3_0B); a3 = *((volatile vector float *)(ptrA+3*M+OFFA+4));              \
  SPU_FMA(c0_1B,a00,b0_1B,c0_1B); *((volatile vector float *)(ptrC+48)) = c0_0A;                   \
  SPU_FMA(c1_1B,a10,b0_1B,c1_1B); *((volatile vector float *)(ptrC+M+48)) = c1_0A;                 \
  SPU_FMA(c2_1B,a20,b0_1B,c2_1B); a00 = spu_shuffle(a0, a0, pat0);                                 \
  SPU_FMA(c3_1B,a30,b0_1B,c3_1B); *((volatile vector float *)(ptrC+2*M+48)) = c2_0A;               \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B); *((volatile vector float *)(ptrC+3*M+48)) = c3_0A;               \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B); a10 = spu_shuffle(a1, a1, pat0);                                 \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); *((volatile vector float *)(ptrC+52)) = c0_1A;                   \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); *((volatile vector float *)(ptrC+M+52)) = c1_1A;                 \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); a20 = spu_shuffle(a2, a2, pat0);                                 \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); *((volatile vector float *)(ptrC+2*M+52)) = c2_1A;               \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); *((volatile vector float *)(ptrC+3*M+52)) = c3_1A;               \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); a30 = spu_shuffle(a3, a3, pat0);                                 \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); c0_0A = *((volatile vector float *)(ptrC));                      \
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); c1_0A = *((volatile vector float *)(ptrC+M));                              \
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); a01 = spu_shuffle(a0, a0, pat1);                                 \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); c2_0A = *((volatile vector float *)(ptrC+2*M));                  \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); c3_0A = *((volatile vector float *)(ptrC+3*M));                  \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); a11 = spu_shuffle(a1, a1, pat1);                                 \
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); b0_0A = *((volatile vector float *)(ptrB+4*M+OFFB*M)); \
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); b0_1A = *((volatile vector float *)(ptrB+4*M+OFFB*M+4));         \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); a21 = spu_shuffle(a2, a2, pat1);                                 \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); b1_0A = *((volatile vector float *)(ptrB+5*M+OFFB*M)); \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); b1_1A = *((volatile vector float *)(ptrB+5*M+OFFB*M+4));         \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); a31 = spu_shuffle(a3, a3, pat1);                                 \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); c0_1A = *((volatile vector float *)(ptrC+4));                              \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); c1_1A = *((volatile vector float *)(ptrC+M+4));                  \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); a02 = spu_shuffle(a0, a0, pat2);                                 \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); c2_1A = *((volatile vector float *)(ptrC+2*M+4));                \
  SPU_FMA(c0_0A,a00,b0_0A,c0_0A); c3_1A = *((volatile vector float *)(ptrC+3*M+4));                \
  SPU_FMA(c1_0A,a10,b0_0A,c1_0A); a12 = spu_shuffle(a1, a1, pat2);                                 \
  SPU_FMA(c2_0A,a20,b0_0A,c2_0A); b2_0A = *((volatile vector float *)(ptrB+6*M+OFFB*M)); \
  SPU_FMA(c3_0A,a30,b0_0A,c3_0A); b2_1A = *((volatile vector float *)(ptrB+6*M+OFFB*M+4));         \
  SPU_FMA(c0_1A,a00,b0_1A,c0_1A); a22 = spu_shuffle(a2, a2, pat2);                                 \
  SPU_FMA(c1_1A,a10,b0_1A,c1_1A); b3_0A = *((volatile vector float *)(ptrB+7*M+OFFB*M)); \
  SPU_FMA(c2_1A,a20,b0_1A,c2_1A); b3_1A = *((volatile vector float *)(ptrB+7*M+OFFB*M+4));         \
  SPU_FMA(c3_1A,a30,b0_1A,c3_1A); a32 = spu_shuffle(a3, a3, pat2);                                 \
  SPU_FMA(c0_0A,a01,b1_0A,c0_0A); *((volatile vector float *)(ptrC+56)) = c0_0B;                   \
  SPU_FMA(c1_0A,a11,b1_0A,c1_0A); *((volatile vector float *)(ptrC+M+56)) = c1_0B;                 \
  SPU_FMA(c2_0A,a21,b1_0A,c2_0A); a03 = spu_shuffle(a0, a0, pat3);                                 \
  SPU_FMA(c3_0A,a31,b1_0A,c3_0A); *((volatile vector float *)(ptrC+2*M+56)) = c2_0B;               \
  SPU_FMA(c0_1A,a01,b1_1A,c0_1A); *((volatile vector float *)(ptrC+3*M+56)) = c3_0B;               \
  SPU_FMA(c1_1A,a11,b1_1A,c1_1A); a13 = spu_shuffle(a1, a1, pat3);                                 \
  SPU_FMA(c2_1A,a21,b1_1A,c2_1A); *((volatile vector float *)(ptrC+60)) = c0_1B;                   \
  SPU_FMA(c3_1A,a31,b1_1A,c3_1A); *((volatile vector float *)(ptrC+M+60)) = c1_1B;                 \
  SPU_FMA(c0_0A,a02,b2_0A,c0_0A); a23 = spu_shuffle(a2, a2, pat3);                                 \
  SPU_FMA(c1_0A,a12,b2_0A,c1_0A); *((volatile vector float *)(ptrC+2*M+60)) = c2_1B;               \
  SPU_FMA(c2_0A,a22,b2_0A,c2_0A); *((volatile vector float *)(ptrC+3*M+60)) = c3_1B;               \
  SPU_FMA(c3_0A,a32,b2_0A,c3_0A); a33 = spu_shuffle(a3, a3, pat3);                                 \
  SPU_FMA(c0_1A,a02,b2_1A,c0_1A); b0_0B = *((volatile vector float *)(ptrB+4*M+OFFB*M+8));         \
  SPU_FMA(c1_1A,a12,b2_1A,c1_1A); b0_1B = *((volatile vector float *)(ptrB+4*M+OFFB*M+12));        \
  SPU_FMA(c2_1A,a22,b2_1A,c2_1A); b1_0B = *((volatile vector float *)(ptrB+5*M+OFFB*M+8));         \
  SPU_FMA(c3_1A,a32,b2_1A,c3_1A); b1_1B = *((volatile vector float *)(ptrB+5*M+OFFB*M+12));        \
  SPU_FMA(c0_0A,a03,b3_0A,c0_0A); c0_0B = *((volatile vector float *)(ptrC+8));                              \
  SPU_FMA(c1_0A,a13,b3_0A,c1_0A); c1_0B = *((volatile vector float *)(ptrC+M+8));                  \
  SPU_FMA(c2_0A,a23,b3_0A,c2_0A); c2_0B = *((volatile vector float *)(ptrC+2*M+8));                \
  SPU_FMA(c3_0A,a33,b3_0A,c3_0A); c3_0B = *((volatile vector float *)(ptrC+3*M+8));                \
  SPU_FMA(c0_1A,a03,b3_1A,c0_1A); c0_1B = *((volatile vector float *)(ptrC+12));                   \
  SPU_FMA(c1_1A,a13,b3_1A,c1_1A); c1_1B = *((volatile vector float *)(ptrC+M+12));                 \
  SPU_FMA(c2_1A,a23,b3_1A,c2_1A); c2_1B = *((volatile vector float *)(ptrC+2*M+12));               \
  SPU_FMA(c3_1A,a33,b3_1A,c3_1A); c3_1B = *((volatile vector float *)(ptrC+3*M+12));               \
}

#define StageMISCmod(OFFA,OFFB)                                                                                \
{                                                                                                                      \
  SPU_FMA(c0_0B,a00,b0_0B,c0_0B); a0 = *((volatile vector float *)(ptrA+OFFA+4));                  \
  SPU_FMA(c1_0B,a10,b0_0B,c1_0B); a1 = *((volatile vector float *)(ptrA+M+OFFA+4));                \
  SPU_FMA(c2_0B,a20,b0_0B,c2_0B); a2 = *((volatile vector float *)(ptrA+2*M+OFFA+4));              \
  SPU_FMA(c3_0B,a30,b0_0B,c3_0B); a3 = *((volatile vector float *)(ptrA+3*M+OFFA+4));              \
  SPU_FMA(c0_1B,a00,b0_1B,c0_1B); *((volatile vector float *)(ptrC+48)) = c0_0A;                   \
  SPU_FMA(c1_1B,a10,b0_1B,c1_1B); *((volatile vector float *)(ptrC+M+48)) = c1_0A;                 \
  SPU_FMA(c2_1B,a20,b0_1B,c2_1B); a00 = spu_shuffle(a0, a0, pat0);                                 \
  SPU_FMA(c3_1B,a30,b0_1B,c3_1B); *((volatile vector float *)(ptrC+2*M+48)) = c2_0A;               \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B); *((volatile vector float *)(ptrC+3*M+48)) = c3_0A;               \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B); a10 = spu_shuffle(a1, a1, pat0);                                 \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); *((volatile vector float *)(ptrC+52)) = c0_1A;                   \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); *((volatile vector float *)(ptrC+M+52)) = c1_1A;                 \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); a20 = spu_shuffle(a2, a2, pat0);                                 \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); *((volatile vector float *)(ptrC+2*M+52)) = c2_1A;               \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); *((volatile vector float *)(ptrC+3*M+52)) = c3_1A;               \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); a30 = spu_shuffle(a3, a3, pat0);                                 \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); c0_0A = *((volatile vector float *)(ptrC));                      \
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); c1_0A = *((volatile vector float *)(ptrC+M));                              \
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); a01 = spu_shuffle(a0, a0, pat1);                                 \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); c2_0A = *((volatile vector float *)(ptrC+2*M));                  \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); c3_0A = *((volatile vector float *)(ptrC+3*M));                  \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); a11 = spu_shuffle(a1, a1, pat1);                                 \
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); b0_0A = *((volatile vector float *)(ptrB+4*M+OFFB*M)); \
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); b0_1A = *((volatile vector float *)(ptrB+4*M+OFFB*M+4));         \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); a21 = spu_shuffle(a2, a2, pat1);                                 \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); b1_0A = *((volatile vector float *)(ptrB+5*M+OFFB*M)); \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); b1_1A = *((volatile vector float *)(ptrB+5*M+OFFB*M+4));         \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); a31 = spu_shuffle(a3, a3, pat1);                                 \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); c0_1A = *((volatile vector float *)(ptrC+4));                              \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); c1_1A = *((volatile vector float *)(ptrC+M+4));                  \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); a02 = spu_shuffle(a0, a0, pat2);                                 \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); c2_1A = *((volatile vector float *)(ptrC+2*M+4));                \
  SPU_FMA(c0_0A,a00,b0_0A,c0_0A); c3_1A = *((volatile vector float *)(ptrC+3*M+4));                \
  SPU_FMA(c1_0A,a10,b0_0A,c1_0A); a12 = spu_shuffle(a1, a1, pat2);                                 \
  SPU_FMA(c2_0A,a20,b0_0A,c2_0A); b2_0A = *((volatile vector float *)(ptrB+6*M+OFFB*M)); \
  SPU_FMA(c3_0A,a30,b0_0A,c3_0A); b2_1A = *((volatile vector float *)(ptrB+6*M+OFFB*M+4));         \
  SPU_FMA(c0_1A,a00,b0_1A,c0_1A); a22 = spu_shuffle(a2, a2, pat2);                                 \
  SPU_FMA(c1_1A,a10,b0_1A,c1_1A); b3_0A = *((volatile vector float *)(ptrB+7*M+OFFB*M)); \
  SPU_FMA(c2_1A,a20,b0_1A,c2_1A); b3_1A = *((volatile vector float *)(ptrB+7*M+OFFB*M+4));         \
  SPU_FMA(c3_1A,a30,b0_1A,c3_1A); a32 = spu_shuffle(a3, a3, pat2);                                 \
  SPU_FMA(c0_0A,a01,b1_0A,c0_0A); *((volatile vector float *)(ptrC+56)) = c0_0B;                   \
  SPU_FMA(c1_0A,a11,b1_0A,c1_0A); *((volatile vector float *)(ptrC+M+56)) = c1_0B;                 \
  SPU_FMA(c2_0A,a21,b1_0A,c2_0A); a03 = spu_shuffle(a0, a0, pat3);                                 \
  SPU_FMA(c3_0A,a31,b1_0A,c3_0A); *((volatile vector float *)(ptrC+2*M+56)) = c2_0B;               \
  SPU_FMA(c0_1A,a01,b1_1A,c0_1A); *((volatile vector float *)(ptrC+3*M+56)) = c3_0B;               \
  SPU_FMA(c1_1A,a11,b1_1A,c1_1A); a13 = spu_shuffle(a1, a1, pat3);                                 \
  SPU_FMA(c2_1A,a21,b1_1A,c2_1A); *((volatile vector float *)(ptrC+60)) = c0_1B;                   \
  SPU_FMA(c3_1A,a31,b1_1A,c3_1A); *((volatile vector float *)(ptrC+M+60)) = c1_1B;                 \
  SPU_FMA(c0_0A,a02,b2_0A,c0_0A); a23 = spu_shuffle(a2, a2, pat3);                                 \
  SPU_FMA(c1_0A,a12,b2_0A,c1_0A); *((volatile vector float *)(ptrC+2*M+60)) = c2_1B;               \
  SPU_FMA(c2_0A,a22,b2_0A,c2_0A); *((volatile vector float *)(ptrC+3*M+60)) = c3_1B;               \
  SPU_FMA(c3_0A,a32,b2_0A,c3_0A); a33 = spu_shuffle(a3, a3, pat3);                                 \
  SPU_FMA(c0_1A,a02,b2_1A,c0_1A); b0_0B = *((volatile vector float *)(ptrB+4*M+OFFB*M+8));         \
  SPU_FMA(c1_1A,a12,b2_1A,c1_1A); b0_1B = *((volatile vector float *)(ptrB+4*M+OFFB*M+12));        \
  SPU_FMA(c2_1A,a22,b2_1A,c2_1A); b1_0B = *((volatile vector float *)(ptrB+5*M+OFFB*M+8));         \
  SPU_FMA(c3_1A,a32,b2_1A,c3_1A); b1_1B = *((volatile vector float *)(ptrB+5*M+OFFB*M+12));        \
  SPU_FMA(c0_0A,a03,b3_0A,c0_0A); c0_0B = *((volatile vector float *)(ptrC+8));                              \
  SPU_FMA(c1_0A,a13,b3_0A,c1_0A); c1_0B = *((volatile vector float *)(ptrC+M+8));                  \
  SPU_FMA(c2_0A,a23,b3_0A,c2_0A); c2_0B = *((volatile vector float *)(ptrC+2*M+8));                \
  SPU_FMA(c3_0A,a33,b3_0A,c3_0A); c3_0B = *((volatile vector float *)(ptrC+3*M+8));                \
  SPU_FMA(c0_1A,a03,b3_1A,c0_1A); b2_0B = *((volatile vector float *)(ptrB+6*M+OFFB*M+8));         \
  SPU_FMA(c1_1A,a13,b3_1A,c1_1A); b2_1B = *((volatile vector float *)(ptrB+6*M+OFFB*M+12));        \
  SPU_FMA(c2_1A,a23,b3_1A,c2_1A); b3_0B = *((volatile vector float *)(ptrB+7*M+OFFB*M+8));         \
  SPU_FMA(c3_1A,a33,b3_1A,c3_1A); b3_1B = *((volatile vector float *)(ptrB+7*M+OFFB*M+12));        \
  ALIGN8B;                                                                                                             \
  c0_1B = *((volatile vector float *)(ptrC+12));                                                   \
  c1_1B = *((volatile vector float *)(ptrC+M+12));                                                           \
  c2_1B = *((volatile vector float *)(ptrC+2*M+12));                                                         \
  c3_1B = *((volatile vector float *)(ptrC+3*M+12));                                                         \
  ptrB += OFFB*M;                                                                                            \
  ALIGN8B;                                                                                                             \
}

#define StageMISCclr()                                                                                       \
{                                                                                                            \
  SPU_FM(c0_0B,a00,b0_0B);        a0 = *((volatile vector float *)(ptrA+4));             \
  SPU_FM(c1_0B,a10,b0_0B);        a1 = *((volatile vector float *)(ptrA+M+4));           \
  SPU_FM(c2_0B,a20,b0_0B);        a2 = *((volatile vector float *)(ptrA+2*M+4));         \
  SPU_FM(c3_0B,a30,b0_0B);        a3 = *((volatile vector float *)(ptrA+3*M+4));         \
  SPU_FM(c0_1B,a00,b0_1B);        *((volatile vector float *)(ptrC+48)) = c0_0A;         \
  SPU_FM(c1_1B,a10,b0_1B);        *((volatile vector float *)(ptrC+M+48)) = c1_0A;       \
  SPU_FM(c2_1B,a20,b0_1B);        a00 = spu_shuffle(a0, a0, pat0);                       \
  SPU_FM(c3_1B,a30,b0_1B);        *((volatile vector float *)(ptrC+2*M+48)) = c2_0A;     \
  SPU_FMA(c0_0B,a01,b1_0B,c0_0B); *((volatile vector float *)(ptrC+3*M+48)) = c3_0A;     \
  SPU_FMA(c1_0B,a11,b1_0B,c1_0B); a10 = spu_shuffle(a1, a1, pat0);                       \
  SPU_FMA(c2_0B,a21,b1_0B,c2_0B); *((volatile vector float *)(ptrC+52)) = c0_1A;         \
  SPU_FMA(c3_0B,a31,b1_0B,c3_0B); *((volatile vector float *)(ptrC+M+52)) = c1_1A;       \
  SPU_FMA(c0_1B,a01,b1_1B,c0_1B); a20 = spu_shuffle(a2, a2, pat0);                       \
  SPU_FMA(c1_1B,a11,b1_1B,c1_1B); *((volatile vector float *)(ptrC+2*M+52)) = c2_1A;     \
  SPU_FMA(c2_1B,a21,b1_1B,c2_1B); *((volatile vector float *)(ptrC+3*M+52)) = c3_1A;     \
  SPU_FMA(c3_1B,a31,b1_1B,c3_1B); a30 = spu_shuffle(a3, a3, pat0);                       \
  SPU_FMA(c0_0B,a02,b2_0B,c0_0B); c0_0A = *((volatile vector float *)(ptrC));            \
  SPU_FMA(c1_0B,a12,b2_0B,c1_0B); c1_0A = *((volatile vector float *)(ptrC+M));                    \
  SPU_FMA(c2_0B,a22,b2_0B,c2_0B); a01 = spu_shuffle(a0, a0, pat1);                       \
  SPU_FMA(c3_0B,a32,b2_0B,c3_0B); c2_0A = *((volatile vector float *)(ptrC+2*M));        \
  SPU_FMA(c0_1B,a02,b2_1B,c0_1B); c3_0A = *((volatile vector float *)(ptrC+3*M));        \
  SPU_FMA(c1_1B,a12,b2_1B,c1_1B); a11 = spu_shuffle(a1, a1, pat1);                       \
  SPU_FMA(c2_1B,a22,b2_1B,c2_1B); b0_0A = *((volatile vector float *)(ptrB+4*M));        \
  SPU_FMA(c3_1B,a32,b2_1B,c3_1B); b0_1A = *((volatile vector float *)(ptrB+4*M+4));      \
  SPU_FMA(c0_0B,a03,b3_0B,c0_0B); a21 = spu_shuffle(a2, a2, pat1);                       \
  SPU_FMA(c1_0B,a13,b3_0B,c1_0B); b1_0A = *((volatile vector float *)(ptrB+5*M));        \
  SPU_FMA(c2_0B,a23,b3_0B,c2_0B); b1_1A = *((volatile vector float *)(ptrB+5*M+4));      \
  SPU_FMA(c3_0B,a33,b3_0B,c3_0B); a31 = spu_shuffle(a3, a3, pat1);                       \
  SPU_FMA(c0_1B,a03,b3_1B,c0_1B); c0_1A = *((volatile vector float *)(ptrC+4));                    \
  SPU_FMA(c1_1B,a13,b3_1B,c1_1B); c1_1A = *((volatile vector float *)(ptrC+M+4));        \
  SPU_FMA(c2_1B,a23,b3_1B,c2_1B); a02 = spu_shuffle(a0, a0, pat2);                       \
  SPU_FMA(c3_1B,a33,b3_1B,c3_1B); c2_1A = *((volatile vector float *)(ptrC+2*M+4));      \
  SPU_FMA(c0_0A,a00,b0_0A,c0_0A); c3_1A = *((volatile vector float *)(ptrC+3*M+4));      \
  SPU_FMA(c1_0A,a10,b0_0A,c1_0A); a12 = spu_shuffle(a1, a1, pat2);                       \
  SPU_FMA(c2_0A,a20,b0_0A,c2_0A); b2_0A = *((volatile vector float *)(ptrB+6*M));        \
  SPU_FMA(c3_0A,a30,b0_0A,c3_0A); b2_1A = *((volatile vector float *)(ptrB+6*M+4));      \
  SPU_FMA(c0_1A,a00,b0_1A,c0_1A); a22 = spu_shuffle(a2, a2, pat2);                       \
  SPU_FMA(c1_1A,a10,b0_1A,c1_1A); b3_0A = *((volatile vector float *)(ptrB+7*M));        \
  SPU_FMA(c2_1A,a20,b0_1A,c2_1A); b3_1A = *((volatile vector float *)(ptrB+7*M+4));      \
  SPU_FMA(c3_1A,a30,b0_1A,c3_1A); a32 = spu_shuffle(a3, a3, pat2);                       \
  SPU_FMA(c0_0A,a01,b1_0A,c0_0A); *((volatile vector float *)(ptrC+56)) = c0_0B;         \
  SPU_FMA(c1_0A,a11,b1_0A,c1_0A); *((volatile vector float *)(ptrC+M+56)) = c1_0B;       \
  SPU_FMA(c2_0A,a21,b1_0A,c2_0A); a03 = spu_shuffle(a0, a0, pat3);                       \
  SPU_FMA(c3_0A,a31,b1_0A,c3_0A); *((volatile vector float *)(ptrC+2*M+56)) = c2_0B;     \
  SPU_FMA(c0_1A,a01,b1_1A,c0_1A); *((volatile vector float *)(ptrC+3*M+56)) = c3_0B;     \
  SPU_FMA(c1_1A,a11,b1_1A,c1_1A); a13 = spu_shuffle(a1, a1, pat3);                       \
  SPU_FMA(c2_1A,a21,b1_1A,c2_1A); *((volatile vector float *)(ptrC+60)) = c0_1B;         \
  SPU_FMA(c3_1A,a31,b1_1A,c3_1A); *((volatile vector float *)(ptrC+M+60)) = c1_1B;       \
  SPU_FMA(c0_0A,a02,b2_0A,c0_0A); a23 = spu_shuffle(a2, a2, pat3);                       \
  SPU_FMA(c1_0A,a12,b2_0A,c1_0A); *((volatile vector float *)(ptrC+2*M+60)) = c2_1B;     \
  SPU_FMA(c2_0A,a22,b2_0A,c2_0A); *((volatile vector float *)(ptrC+3*M+60)) = c3_1B;   \
  SPU_FMA(c3_0A,a32,b2_0A,c3_0A); a33 = spu_shuffle(a3, a3, pat3);                      \
  SPU_FMA(c0_1A,a02,b2_1A,c0_1A); b0_0B = *((volatile vector float *)(ptrB+4*M+8));     \
  SPU_FMA(c1_1A,a12,b2_1A,c1_1A); b0_1B = *((volatile vector float *)(ptrB+4*M+12));    \
  SPU_FMA(c2_1A,a22,b2_1A,c2_1A); b1_0B = *((volatile vector float *)(ptrB+5*M+8));     \
  SPU_FMA(c3_1A,a32,b2_1A,c3_1A); b1_1B = *((volatile vector float *)(ptrB+5*M+12));    \
  SPU_FMA(c0_0A,a03,b3_0A,c0_0A); c0_0B = *((volatile vector float *)(ptrC+8));                   \
  SPU_FMA(c1_0A,a13,b3_0A,c1_0A); c1_0B = *((volatile vector float *)(ptrC+M+8));       \
  SPU_FMA(c2_0A,a23,b3_0A,c2_0A); c2_0B = *((volatile vector float *)(ptrC+2*M+8));     \
  SPU_FMA(c3_0A,a33,b3_0A,c3_0A); c3_0B = *((volatile vector float *)(ptrC+3*M+8));     \
  SPU_FMA(c0_1A,a03,b3_1A,c0_1A); b2_0B = *((volatile vector float *)(ptrB+6*M+8));     \
  SPU_FMA(c1_1A,a13,b3_1A,c1_1A); b2_1B = *((volatile vector float *)(ptrB+6*M+12));    \
  SPU_FMA(c2_1A,a23,b3_1A,c2_1A); b3_0B = *((volatile vector float *)(ptrB+7*M+8));     \
  SPU_FMA(c3_1A,a33,b3_1A,c3_1A); b3_1B = *((volatile vector float *)(ptrB+7*M+12));    \
  ALIGN8B;                                                                                                  \
  c0_1B = *((volatile vector float *)(ptrC+12));                                        \
  c1_1B = *((volatile vector float *)(ptrC+M+12));                                                \
  c2_1B = *((volatile vector float *)(ptrC+2*M+12));                                              \
  c3_1B = *((volatile vector float *)(ptrC+3*M+12));                                              \
}

#define Loads4RegSetA(OFFSET)                              \
{                                                                    \
  c0_0A = *((volatile vector float *)(ptrC+OFFSET));                 \
  c1_0A = *((volatile vector float *)(ptrC+M+OFFSET));               \
  c2_0A = *((volatile vector float *)(ptrC+2*M+OFFSET));   \
  c3_0A = *((volatile vector float *)(ptrC+3*M+OFFSET));   \
  c0_1A = *((volatile vector float *)(ptrC+OFFSET+4));               \
  c1_1A = *((volatile vector float *)(ptrC+M+OFFSET+4));   \
  c2_1A = *((volatile vector float *)(ptrC+2*M+OFFSET+4)); \
  c3_1A = *((volatile vector float *)(ptrC+3*M+OFFSET+4)); \
  b0_0A = *((volatile vector float *)(ptrB+OFFSET));                 \
  b1_0A = *((volatile vector float *)(ptrB+M+OFFSET));               \
  b2_0A = *((volatile vector float *)(ptrB+2*M+OFFSET));   \
  b3_0A = *((volatile vector float *)(ptrB+3*M+OFFSET));   \
  b0_1A = *((volatile vector float *)(ptrB+OFFSET+4));               \
  b1_1A = *((volatile vector float *)(ptrB+M+OFFSET+4));   \
  b2_1A = *((volatile vector float *)(ptrB+2*M+OFFSET+4)); \
  b3_1A = *((volatile vector float *)(ptrB+3*M+OFFSET+4)); \
}

#define Loads4RegSetAClr(OFFSET)                           \
{                                                                    \
  b0_0A = *((volatile vector float *)(ptrB+OFFSET));                 \
  b1_0A = *((volatile vector float *)(ptrB+M+OFFSET));               \
  b2_0A = *((volatile vector float *)(ptrB+2*M+OFFSET));   \
  b3_0A = *((volatile vector float *)(ptrB+3*M+OFFSET));   \
  b0_1A = *((volatile vector float *)(ptrB+OFFSET+4));               \
  b1_1A = *((volatile vector float *)(ptrB+M+OFFSET+4));   \
  b2_1A = *((volatile vector float *)(ptrB+2*M+OFFSET+4)); \
  b3_1A = *((volatile vector float *)(ptrB+3*M+OFFSET+4)); \
}

#define Ops4RegSetAClr()                         \
{                                                          \
  c0_0A = spu_mul( a00, b0_0A);                            \
  c1_0A = spu_mul( a10, b0_0A);                            \
  c2_0A = spu_mul( a20, b0_0A);                            \
  c3_0A = spu_mul( a30, b0_0A);                            \
  c0_1A = spu_mul( a00, b0_1A);                            \
  c1_1A = spu_mul( a10, b0_1A);                            \
  c2_1A = spu_mul( a20, b0_1A);                            \
  c3_1A = spu_mul( a30, b0_1A);                            \
  c0_0A = spu_madd(a01, b1_0A, c0_0A);           \
  c1_0A = spu_madd(a11, b1_0A, c1_0A);           \
  c2_0A = spu_madd(a21, b1_0A, c2_0A);           \
  c3_0A = spu_madd(a31, b1_0A, c3_0A);           \
  c0_1A = spu_madd(a01, b1_1A, c0_1A);           \
  c1_1A = spu_madd(a11, b1_1A, c1_1A);           \
  c2_1A = spu_madd(a21, b1_1A, c2_1A);           \
  c3_1A = spu_madd(a31, b1_1A, c3_1A);           \
  c0_0A = spu_madd(a02, b2_0A, c0_0A);           \
  c1_0A = spu_madd(a12, b2_0A, c1_0A);           \
  c2_0A = spu_madd(a22, b2_0A, c2_0A);           \
  c3_0A = spu_madd(a32, b2_0A, c3_0A);           \
  c0_1A = spu_madd(a02, b2_1A, c0_1A);           \
  c1_1A = spu_madd(a12, b2_1A, c1_1A);           \
  c2_1A = spu_madd(a22, b2_1A, c2_1A);           \
  c3_1A = spu_madd(a32, b2_1A, c3_1A);           \
  c0_0A = spu_madd(a03, b3_0A, c0_0A);           \
  c1_0A = spu_madd(a13, b3_0A, c1_0A);           \
  c2_0A = spu_madd(a23, b3_0A, c2_0A);           \
  c3_0A = spu_madd(a33, b3_0A, c3_0A);           \
  c0_1A = spu_madd(a03, b3_1A, c0_1A);           \
  c1_1A = spu_madd(a13, b3_1A, c1_1A);           \
  c2_1A = spu_madd(a23, b3_1A, c2_1A);           \
  c3_1A = spu_madd(a33, b3_1A, c3_1A);           \
}

#define Ops4RegSetA()                                      \
{                                                          \
  c0_0A = spu_madd(a00, b0_0A, c0_0A);           \
  c1_0A = spu_madd(a10, b0_0A, c1_0A);           \
  c2_0A = spu_madd(a20, b0_0A, c2_0A);           \
  c3_0A = spu_madd(a30, b0_0A, c3_0A);           \
  c0_1A = spu_madd(a00, b0_1A, c0_1A);           \
  c1_1A = spu_madd(a10, b0_1A, c1_1A);           \
  c2_1A = spu_madd(a20, b0_1A, c2_1A);           \
  c3_1A = spu_madd(a30, b0_1A, c3_1A);           \
  c0_0A = spu_madd(a01, b1_0A, c0_0A);           \
  c1_0A = spu_madd(a11, b1_0A, c1_0A);           \
  c2_0A = spu_madd(a21, b1_0A, c2_0A);           \
  c3_0A = spu_madd(a31, b1_0A, c3_0A);           \
  c0_1A = spu_madd(a01, b1_1A, c0_1A);           \
  c1_1A = spu_madd(a11, b1_1A, c1_1A);           \
  c2_1A = spu_madd(a21, b1_1A, c2_1A);           \
  c3_1A = spu_madd(a31, b1_1A, c3_1A);           \
  c0_0A = spu_madd(a02, b2_0A, c0_0A);           \
  c1_0A = spu_madd(a12, b2_0A, c1_0A);           \
  c2_0A = spu_madd(a22, b2_0A, c2_0A);           \
  c3_0A = spu_madd(a32, b2_0A, c3_0A);           \
  c0_1A = spu_madd(a02, b2_1A, c0_1A);           \
  c1_1A = spu_madd(a12, b2_1A, c1_1A);           \
  c2_1A = spu_madd(a22, b2_1A, c2_1A);           \
  c3_1A = spu_madd(a32, b2_1A, c3_1A);           \
  c0_0A = spu_madd(a03, b3_0A, c0_0A);           \
  c1_0A = spu_madd(a13, b3_0A, c1_0A);           \
  c2_0A = spu_madd(a23, b3_0A, c2_0A);           \
  c3_0A = spu_madd(a33, b3_0A, c3_0A);           \
  c0_1A = spu_madd(a03, b3_1A, c0_1A);           \
  c1_1A = spu_madd(a13, b3_1A, c1_1A);           \
  c2_1A = spu_madd(a23, b3_1A, c2_1A);           \
  c3_1A = spu_madd(a33, b3_1A, c3_1A);           \
}

#define Stores4RegSetA(OFFSET)                                                  \
{                                                                               \
  *((volatile vector float *)(ptrC+OFFSET)) = c0_0A;                  \
  *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0A;                \
  *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0A;    \
  *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0A;    \
  *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1A;                \
  *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1A;    \
  *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1A; \
  *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1A; \
}

#define Loads4RegSetB(OFFSET)                                         \
{                                                                               \
  c0_0B = *((volatile vector float *)(ptrC+OFFSET));                  \
  c1_0B = *((volatile vector float *)(ptrC+M+OFFSET));                \
  c2_0B = *((volatile vector float *)(ptrC+2*M+OFFSET));    \
  c3_0B = *((volatile vector float *)(ptrC+3*M+OFFSET));    \
  c0_1B = *((volatile vector float *)(ptrC+OFFSET+4));                \
  c1_1B = *((volatile vector float *)(ptrC+M+OFFSET+4));    \
  c2_1B = *((volatile vector float *)(ptrC+2*M+OFFSET+4)); \
  c3_1B = *((volatile vector float *)(ptrC+3*M+OFFSET+4)); \
  b0_0B = *((volatile vector float *)(ptrB+OFFSET));                  \
  b1_0B = *((volatile vector float *)(ptrB+M+OFFSET));                \
  b2_0B = *((volatile vector float *)(ptrB+2*M+OFFSET));    \
  b3_0B = *((volatile vector float *)(ptrB+3*M+OFFSET));    \
  b0_1B = *((volatile vector float *)(ptrB+OFFSET+4));                \
  b1_1B = *((volatile vector float *)(ptrB+M+OFFSET+4));    \
  b2_1B = *((volatile vector float *)(ptrB+2*M+OFFSET+4)); \
  b3_1B = *((volatile vector float *)(ptrB+3*M+OFFSET+4)); \
}

#define Loads4RegSetBClr(OFFSET)                                      \
{                                                                               \
  b0_0B = *((volatile vector float *)(ptrB+OFFSET));                  \
  b1_0B = *((volatile vector float *)(ptrB+M+OFFSET));                \
  b2_0B = *((volatile vector float *)(ptrB+2*M+OFFSET));    \
  b3_0B = *((volatile vector float *)(ptrB+3*M+OFFSET));    \
  b0_1B = *((volatile vector float *)(ptrB+OFFSET+4));                \
  b1_1B = *((volatile vector float *)(ptrB+M+OFFSET+4));    \
  b2_1B = *((volatile vector float *)(ptrB+2*M+OFFSET+4)); \
  b3_1B = *((volatile vector float *)(ptrB+3*M+OFFSET+4)); \
}

#define Ops4RegSetB()                                                 \
{                                                                     \
  c0_0B = spu_madd(a00, b0_0B, c0_0B);                      \
  c1_0B = spu_madd(a10, b0_0B, c1_0B);                      \
  c2_0B = spu_madd(a20, b0_0B, c2_0B);                      \
  c3_0B = spu_madd(a30, b0_0B, c3_0B);                      \
  c0_1B = spu_madd(a00, b0_1B, c0_1B);                      \
  c1_1B = spu_madd(a10, b0_1B, c1_1B);                      \
  c2_1B = spu_madd(a20, b0_1B, c2_1B);                      \
  c3_1B = spu_madd(a30, b0_1B, c3_1B);                      \
  c0_0B = spu_madd(a01, b1_0B, c0_0B);                      \
  c1_0B = spu_madd(a11, b1_0B, c1_0B);                      \
  c2_0B = spu_madd(a21, b1_0B, c2_0B);                      \
  c3_0B = spu_madd(a31, b1_0B, c3_0B);                      \
  c0_1B = spu_madd(a01, b1_1B, c0_1B);                      \
  c1_1B = spu_madd(a11, b1_1B, c1_1B);                      \
  c2_1B = spu_madd(a21, b1_1B, c2_1B);                      \
  c3_1B = spu_madd(a31, b1_1B, c3_1B);                      \
  c0_0B = spu_madd(a02, b2_0B, c0_0B); SPU_LNOP; \
  c1_0B = spu_madd(a12, b2_0B, c1_0B);                      \
  c2_0B = spu_madd(a22, b2_0B, c2_0B);                      \
  c3_0B = spu_madd(a32, b2_0B, c3_0B);                      \
  c0_1B = spu_madd(a02, b2_1B, c0_1B);                      \
  c1_1B = spu_madd(a12, b2_1B, c1_1B);                      \
  c2_1B = spu_madd(a22, b2_1B, c2_1B);                      \
  c3_1B = spu_madd(a32, b2_1B, c3_1B);                      \
  c0_0B = spu_madd(a03, b3_0B, c0_0B);                      \
  c1_0B = spu_madd(a13, b3_0B, c1_0B);                      \
  c2_0B = spu_madd(a23, b3_0B, c2_0B);                      \
  c3_0B = spu_madd(a33, b3_0B, c3_0B);                      \
  c0_1B = spu_madd(a03, b3_1B, c0_1B);                      \
  c1_1B = spu_madd(a13, b3_1B, c1_1B);                      \
  c2_1B = spu_madd(a23, b3_1B, c2_1B);                      \
  c3_1B = spu_madd(a33, b3_1B, c3_1B);                      \
}

#define Stores4RegSetB(OFFSET)                                                  \
{                                                                              \
  *((volatile vector float *)(ptrC+OFFSET)) = c0_0B;                \
  *((volatile vector float *)(ptrC+M+OFFSET)) = c1_0B;              \
  *((volatile vector float *)(ptrC+2*M+OFFSET)) = c2_0B;   \
  *((volatile vector float *)(ptrC+3*M+OFFSET)) = c3_0B;   \
  *((volatile vector float *)(ptrC+OFFSET+4)) = c0_1B;              \
  *((volatile vector float *)(ptrC+M+OFFSET+4)) = c1_1B;   \
  *((volatile vector float *)(ptrC+2*M+OFFSET+4)) = c2_1B; \
  *((volatile vector float *)(ptrC+3*M+OFFSET+4)) = c3_1B; \
}

float blkA0[M*M] __attribute__((aligned(128)));
float blkB0[M*M] __attribute__((aligned(128)));
float blkC0[M*M] __attribute__((aligned(128)));
float blkA1[M*M] __attribute__((aligned(128)));
float blkB1[M*M] __attribute__((aligned(128)));
float blkC1[M*M] __attribute__((aligned(128)));

// For synchronnization
char lsbuf[128] __attribute__((aligned(128)));


static void Block_Fetch(volatile float *blkA, volatile float *blkB,
                              unsigned int by, unsigned int bx, unsigned int idx, unsigned int rtag)
{
  unsigned int baseA, baseB;

  baseA = A_AREA + 4*(by*M*N+idx*M*M);
  baseB = B_AREA + 4*(bx*M*M+idx*M*N);

  mfc_get(blkA, baseA, sizeof(float)*M*M, rtag, 0, 0);
  mfc_get(blkB, baseB, sizeof(float)*M*M, rtag, 0, 0);
}




static void MatInit_MxM(volatile float *blkC, volatile float *blkA, volatile float *blkB)
{
  unsigned int i;
  volatile float *ptrA, *ptrB, *ptrC;

  vector float a0, a1, a2, a3;

  vector float a00, a01, a02, a03;
  vector float a10, a11, a12, a13;
  vector float a20, a21, a22, a23;
  vector float a30, a31, a32, a33;

  vector float b0_0A, b1_0A, b2_0A, b3_0A;
  vector float c0_0A, c1_0A, c2_0A, c3_0A;
  vector float b0_1A, b1_1A, b2_1A, b3_1A;
  vector float c0_1A, c1_1A, c2_1A, c3_1A;

  vector float b0_0B, b1_0B, b2_0B, b3_0B;
  vector float c0_0B, c1_0B, c2_0B, c3_0B;
  vector float b0_1B, b1_1B, b2_1B, b3_1B;
  vector float c0_1B, c1_1B, c2_1B, c3_1B;

  vector float b0_0C, b1_0C, b2_0C, b3_0C;
  vector float c0_0C, c1_0C, c2_0C, c3_0C;
  vector float b0_1C, b1_1C, b2_1C, b3_1C;
  vector float c0_1C, c1_1C, c2_1C, c3_1C;

  const vector unsigned char pat0 = VEC_LITERAL(vector unsigned char,
                                                           0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03,
                                                           0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03);

  const vector unsigned char pat1 = VEC_LITERAL(vector unsigned char,
                                                           0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07,
                                                           0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07);

  const vector unsigned char pat2 = VEC_LITERAL(vector unsigned char,
                                                           0x08, 0x09, 0x0a, 0x0b, 0x08, 0x09, 0x0a, 0x0b,
                                                           0x08, 0x09, 0x0a, 0x0b, 0x08, 0x09, 0x0a, 0x0b);

  const vector unsigned char pat3 = VEC_LITERAL(vector unsigned char,
                                                           0x0c, 0x0d, 0x0e, 0x0f, 0x0c, 0x0d, 0x0e, 0x0f,
                                                           0x0c, 0x0d, 0x0e, 0x0f, 0x0c, 0x0d, 0x0e, 0x0f);

  for(i=0; i<M; i+=4){
    ptrA = &blkA[i*M];
    ptrB = &blkB[0];
    ptrC = &blkC[i*M];
    a0 = *((volatile vector float *)(ptrA));
    a1 = *((volatile vector float *)(ptrA+M));
    a2 = *((volatile vector float *)(ptrA+2*M));
    a3 = *((volatile vector float *)(ptrA+3*M));
    a00 = spu_shuffle(a0, a0, pat0);
    a01 = spu_shuffle(a0, a0, pat1);
    a02 = spu_shuffle(a0, a0, pat2);
    a03 = spu_shuffle(a0, a0, pat3);
    a10 = spu_shuffle(a1, a1, pat0);
    a11 = spu_shuffle(a1, a1, pat1);
    a12 = spu_shuffle(a1, a1, pat2);
    a13 = spu_shuffle(a1, a1, pat3);
    a20 = spu_shuffle(a2, a2, pat0);
    a21 = spu_shuffle(a2, a2, pat1);
    a22 = spu_shuffle(a2, a2, pat2);
    a23 = spu_shuffle(a2, a2, pat3);
    a30 = spu_shuffle(a3, a3, pat0);
    a31 = spu_shuffle(a3, a3, pat1);
    a32 = spu_shuffle(a3, a3, pat2);
```

```
a33 = spu_shuffle(a3, a3, pat3);
Loads4RegSetAClr(0);
Ops4RegSetAClr();
Loads4RegSetBClr(8);
StageCBAclr(0);
StageACBclr(8);
StageBACclr(16);
StageCBAclr(24);
StageACBclr(32);
StageBACclr(40);
StageMISCclr();
StageCBA(0,4);
StageACB(8,4);
StageBAC(16,4);
StageCBA(24,4);
StageACB(32,4);
StageBAC(40,4);
StageMISC(4,4);
StageCBAmod(0,8);
StageACB(8,8);
StageBAC(16,8);
StageCBA(24,8);
StageACB(32,8);
StageBAC(40,8);
StageMISC(8,8);
StageCBAmod(0,12);
StageACB(8,12);
StageBAC(16,12);
StageCBA(24,12);
StageACB(32,12);
StageBAC(40,12);
StageMISC(12,12);
StageCBAmod(0,16);
StageACB(8,16);
StageBAC(16,16);
StageCBA(24,16);
StageACB(32,16);
StageBAC(40,16);
StageMISC(16,16);
StageCBAmod(0,20);
StageACB(8,20);
StageBAC(16,20);
StageCBA(24,20);
StageACB(32,20);
StageBAC(40,20);
StageMISC(20,20);
StageCBAmod(0,24);
StageACB(8,24);
StageBAC(16,24);
StageCBA(24,24);
StageACB(32,24);
StageBAC(40,24);
StageMISCmod(24,24);
StageCBA(0,4);
StageACB(8,4);
StageBAC(16,4);
StageCBA(24,4);
StageACB(32,4);
StageBAC(40,4);
StageMISC(28,4);
StageCBAmod(0,8);
StageACB(8,8);
StageBAC(16,8);
StageCBA(24,8);
StageACB(32,8);
StageBAC(40,8);
StageMISC(32,8);
StageCBAmod(0,12);
StageACB(8,12);
StageBAC(16,12);
StageCBA(24,12);
StageACB(32,12);
StageBAC(40,12);
StageMISC(36,12);
StageCBAmod(0,16);
StageACB(8,16);
StageBAC(16,16);
StageCBA(24,16);
StageACB(32,16);
StageBAC(40,16);
StageMISC(40,16);
StageCBAmod(0,20);
StageACB(8,20);
StageBAC(16,20);
StageCBA(24,20);
StageACB(32,20);
StageBAC(40,20);
StageMISC(44,20);
StageCBAmod(0,24);
StageACB(8,24);
StageBAC(16,24);
StageCBA(24,24);
StageACB(32,24);
StageBAC(40,24);
StageMISCmod(48,24);
StageCBA(0,4);
StageACB(8,4);
StageBAC(16,4);
StageCBA(24,4);
StageACB(32,4);
StageBAC(40,4);
StageMISC(52,4);
StageCBAmod(0,8);
```

```
    StageACB(8,8);
    StageBAC(16,8);
    StageCBA(24,8);
    StageACB(32,8);
    StageBAC(40,8);
    StageMISC(56,8);
    StageCBAmod(0,12);
    StageACB(8,12);
    StageBAC(16,12);
    StageCBA(24,12);
    StageACB(32,12);
    StageBAC(40,12);
    Ops4RegSetB();
    Stores4RegSetA(48);
    Stores4RegSetB(56);
  }
}

static void MatMult_MxM(volatile float *blkC, volatile float *blkA, volatile float *blkB)
{
  unsigned int i;
  volatile float *ptrA, *ptrB, *ptrC;

  vector float a0, a1, a2, a3;

  vector float a00, a01, a02, a03;
  vector float a10, a11, a12, a13;
  vector float a20, a21, a22, a23;
  vector float a30, a31, a32, a33;

  vector float b0_0A, b1_0A, b2_0A, b3_0A;
  vector float c0_0A, c1_0A, c2_0A, c3_0A;
  vector float b0_1A, b1_1A, b2_1A, b3_1A;
  vector float c0_1A, c1_1A, c2_1A, c3_1A;

  vector float b0_0B, b1_0B, b2_0B, b3_0B;
  vector float c0_0B, c1_0B, c2_0B, c3_0B;
  vector float b0_1B, b1_1B, b2_1B, b3_1B;
  vector float c0_1B, c1_1B, c2_1B, c3_1B;

  vector float b0_0C, b1_0C, b2_0C, b3_0C;
  vector float c0_0C, c1_0C, c2_0C, c3_0C;
  vector float b0_1C, b1_1C, b2_1C, b3_1C;
  vector float c0_1C, c1_1C, c2_1C, c3_1C;

  const vector unsigned char pat0 = VEC_LITERAL(vector unsigned char,
                                                           0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03,
                                                           0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03);

  const vector unsigned char pat1 = VEC_LITERAL(vector unsigned char,
                                                           0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07,
                                                           0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07);

  const vector unsigned char pat2 = VEC_LITERAL(vector unsigned char,
                                                           0x08, 0x09, 0x0a, 0x0b, 0x08, 0x09, 0x0a, 0x0b,
                                                           0x08, 0x09, 0x0a, 0x0b, 0x08, 0x09, 0x0a, 0x0b);

  const vector unsigned char pat3 = VEC_LITERAL(vector unsigned char,
                                                           0x0c, 0x0d, 0x0e, 0x0f, 0x0c, 0x0d, 0x0e, 0x0f,
                                                           0x0c, 0x0d, 0x0e, 0x0f, 0x0c, 0x0d, 0x0e, 0x0f);

  for(i=0; i<M; i+=4){
    ptrA = &blkA[i*M];
    ptrB = &blkB[0];
    ptrC = &blkC[i*M];
    a0 = *((volatile vector float *)(ptrA));
    a1 = *((volatile vector float *)(ptrA+M));
    a2 = *((volatile vector float *)(ptrA+2*M));
    a3 = *((volatile vector float *)(ptrA+3*M));
    a00 = spu_shuffle(a0, a0, pat0);
    a01 = spu_shuffle(a0, a0, pat1);
    a02 = spu_shuffle(a0, a0, pat2);
    a03 = spu_shuffle(a0, a0, pat3);
    a10 = spu_shuffle(a1, a1, pat0);
    a11 = spu_shuffle(a1, a1, pat1);
    a12 = spu_shuffle(a1, a1, pat2);
    a13 = spu_shuffle(a1, a1, pat3);
    a20 = spu_shuffle(a2, a2, pat0);
    a21 = spu_shuffle(a2, a2, pat1);
    a22 = spu_shuffle(a2, a2, pat2);
    a23 = spu_shuffle(a2, a2, pat3);
    a30 = spu_shuffle(a3, a3, pat0);
    a31 = spu_shuffle(a3, a3, pat1);
    a32 = spu_shuffle(a3, a3, pat2);
    a33 = spu_shuffle(a3, a3, pat3);
    Loads4RegSetA(0);
    Ops4RegSetA();
    Loads4RegSetB(8);
    StageCBA(0,0);
    StageACB(8,0);
    StageBAC(16,0);
    StageCBA(24,0);
    StageACB(32,0);
    StageBAC(40,0);
    StageMISC(0,0);
    StageCBAmod(0,4);
    StageACB(8,4);
    StageBAC(16,4);
    StageCBA(24,4);
    StageACB(32,4);
    StageBAC(40,4);
    StageMISC(4,4);
    StageCBAmod(0,8);
    StageACB(8,8);
    StageBAC(16,8);
    StageCBA(24,8);
    StageACB(32,8);
    StageBAC(40,8);
    StageMISC(8,8);
    StageCBAmod(0,12);
    StageACB(8,12);
    StageBAC(16,12);
    StageCBA(24,12);
    StageACB(32,12);
    StageBAC(40,12);
    StageMISC(12,12);
    StageCBAmod(0,16);
    StageACB(8,16);
    StageBAC(16,16);
    StageCBA(24,16);
    StageACB(32,16);
    StageBAC(40,16);
    StageMISC(16,16);
    StageCBAmod(0,20);
    StageACB(8,20);
    StageBAC(16,20);
    StageCBA(24,20);
    StageACB(32,20);
    StageBAC(40,20);
    StageMISC(20,20);
    StageCBAmod(0,24);
    StageACB(8,24);
    StageBAC(16,24);
    StageCBA(24,24);
    StageACB(32,24);
    StageBAC(40,24);
    StageMISCmod(24,24);
    StageCBA(0,4);
    StageACB(8,4);
    StageBAC(16,4);
    StageCBA(24,4);
    StageACB(32,4);
    StageBAC(40,4);
    StageMISC(28,4);
    StageCBAmod(0,8);
    StageACB(8,8);
    StageBAC(16,8);
    StageCBA(24,8);
    StageACB(32,8);
    StageBAC(40,8);
    StageMISC(32,8);
    StageCBAmod(0,12);
    StageACB(8,12);
    StageBAC(16,12);
    StageCBA(24,12);
    StageACB(32,12);
    StageBAC(40,12);
    StageMISC(36,12);
    StageCBAmod(0,16);
    StageACB(8,16);
    StageBAC(16,16);
    StageCBA(24,16);
    StageACB(32,16);
    StageBAC(40,16);
    StageMISC(40,16);
    StageCBAmod(0,20);
    StageACB(8,20);
    StageBAC(16,20);
    StageCBA(24,20);
    StageACB(32,20);
    StageBAC(40,20);
    StageMISC(44,20);
    StageCBAmod(0,24);
    StageACB(8,24);
    StageBAC(16,24);
    StageCBA(24,24);
    StageACB(32,24);
    StageBAC(40,24);
    StageMISCmod(48,24);
    StageCBA(0,4);
    StageACB(8,4);
    StageBAC(16,4);
    StageCBA(24,4);
    StageACB(32,4);
    StageBAC(40,4);
    StageMISC(52,4);
    StageCBAmod(0,8);
    StageACB(8,8);
    StageBAC(16,8);
    StageCBA(24,8);
    StageACB(32,8);
    StageBAC(40,8);
    StageMISC(56,8);
    StageCBAmod(0,12);
    StageACB(8,12);
    StageBAC(16,12);
    StageCBA(24,12);
    StageACB(32,12);
    StageBAC(40,12);
    Ops4RegSetB();
    Stores4RegSetA(48);
    Stores4RegSetB(56);
  }
}
```

```
static void Block_Store(volatile float *blkC, int by, int bx, int wtag)
{
  unsigned int baseC;

  baseC = C_AREA + 4*(by*M*N+bx*M*M);

  mfc_put(blkC, baseC, sizeof(float)*M*M, wtag, 0, 0);
}


// Synchronously acquire the next block to be computed. Synchronize the threads
// when a iterations is complete.
//
// This is accomplished by atomically updating 2 variables - blocks started, and
// blocks completed.
//
// WARNING - This function assumes that blocks is a power of two.

static volatile unsigned int atomic_buffer[32] __attribute__ ((aligned (128)));

static unsigned int Next_Block(unsigned int blocks,        /* blocks per iteration */
                                     unsigned int completed)          /* flag indicating that a block was completed */
{
  unsigned int started, finished;
  unsigned int status, do_store;

  do {
    /* Fetch the atomic block counters.
     */
    do {
      mfc_getllar(atomic_buffer, FINC_AREA, 0, 0);
      status = mfc_read_atomic_status() & MFC_GETLLAR_STATUS;
    } while (status == 0);

    /* Increment the started count if their are anymore in this iteration.
     * Increment the finished count if told that a block was completed.
     */
    started = atomic_buffer[0];
    finished = atomic_buffer[4] + completed;
    atomic_buffer[4] = finished;

    do_store = completed;

    if (started < blocks) {
      atomic_buffer[0] = (started + 1);
      do_store = 1;
    }

    status = 0;

    if (do_store) {
      /* Update the atomic values - started and finished.
       */
      mfc_putllc(atomic_buffer, FINC_AREA, 0, 0);
      status = mfc_read_atomic_status() & MFC_PUTLLC_STATUS;
    }
  } while (status);

  /* Return the block index
   */
  return (started);
}

// Wait for all the blocks of this interation to finish. Also increment the finished count
// if instructed to do so. The last block to be marked completed will also zero the block
// started count.

static void Wait_For_All(unsigned int blocks,              /* wait until the finished count reaches this value */
                               unsigned int completed)      /* flag indicating that a block was completed */
{
  unsigned int finished;
  unsigned int status;

  do {
    /* Fetch the atomic block counters.
     */
    do {
      mfc_getllar(atomic_buffer, FINC_AREA, 0, 0);
      status = mfc_read_atomic_status() & MFC_GETLLAR_STATUS;
    } while (status == 0);

    /* Increment the finished count if told that a block was completed.
     */
    finished =  atomic_buffer[4] + completed;

    status = 0;

    if (completed) {
      /* Update the atomic values.
       */
      atomic_buffer[4] = finished;

      mfc_putllc(atomic_buffer, FINC_AREA, 0, 0);
      status = mfc_read_atomic_status() & MFC_PUTLLC_STATUS;

      if (status == 0) completed = 0;
    }
  } while (status || (finished < blocks));
}



int main()
```

```
{
  unsigned int i, j, k, iter;
  unsigned int i_next, j_next;

  volatile float *InA, *InB;
  volatile float *OpA, *OpB, *OpC;
  volatile float *OutC;

  unsigned int InTag = DIN_TAG;
  unsigned int OutTag = DOUT_TAG;

  unsigned int blkid = 0;
  unsigned int total_blks = 0;
  unsigned int shift, mask;
  unsigned int blocks, tiles;
  unsigned int blk_completed;

  // Double Buffer Initialization
  InA = blkA0;
  InB = blkB0;
  OpA = blkA0;
  OpB = blkB0;
  OpC = blkC0;
  OutC = blkC0;

  // Fetch working parameters from mailbox. The input
  // parameters include:
  //      N = the size of the matrix
  //      ITER = number of times to perform matrix multiply
  //      FINC_AREA = pointer to synchronization counter
  //      A_AREA = pointer to input matrix A
  //      B_AREA = pointer to input matrix B
  //      C_AREA = pointer to output matrix C

  N = spu_read_in_mbox();
  ITER = spu_read_in_mbox();
  FINC_AREA = spu_read_in_mbox();
  A_AREA = spu_read_in_mbox();
  B_AREA = spu_read_in_mbox();
  C_AREA = spu_read_in_mbox();

  tiles = N / M;
  blocks = tiles * tiles;
  mask  = tiles - 1;
  shift = 32 - spu_extract(spu_cntlz(spu_promote(mask, 0)), 0);

  // Matrix Multiply with block partitioning
  for (iter=0; iter<ITER; iter++) {
    total_blks += blocks;

    blkid = Next_Block(total_blks, 0);
    blk_completed = 0;
    i = (blkid >> shift) & mask;
    j = (blkid) & mask;
    if (blkid < total_blks) {
      Block_Fetch(InA, InB, i, j, 0, InTag);

      while (1) {
          SwapInBuf();
          Block_Fetch(InA, InB, i, j, 1, InTag);
          DMA_Wait(InTag^1);
          DMA_Wait(OutTag);
          MatInit_MxM(OpC, OpA, OpB);
          for(k=1;k<(N/M)-1;k++){
            SwapInBuf();
            Block_Fetch(InA, InB, i, j, k+1, InTag);
            DMA_Wait(InTag^1);
            MatMult_MxM(OpC, OpA, OpB);
          }
          // Epilogue for k==(N/M)-1
          SwapInBuf();

          blkid = Next_Block(total_blks, blk_completed);
          blk_completed = 0;
          i_next = (blkid >> shift) & mask;
          j_next = (blkid) & mask;

          if (blkid >= total_blks) {
            DMA_Wait(InTag^1);
            if (k < (N/M)) {
              MatMult_MxM(OpC, OpA, OpB);
            }
            Block_Store(OutC, i, j, OutTag);
            blk_completed++;
            DMA_Wait(OutTag);
            break;
          }

          Block_Fetch(InA, InB, i_next, j_next, 0, InTag);
          DMA_Wait(InTag^1);
          if (k < (N/M)) {
            MatMult_MxM(OpC, OpA, OpB);
          }

          Block_Store(OutC, i, j, OutTag);
          blk_completed++;
          i = i_next;
          j = j_next;
          SwapOutBuf();
      }
    }

    // Do a synchronization between matrices by waiting for all
    // the blocks to be completed by all participating SPEs..
    Wait_For_All(total_blks, blk_completed);
  }

  return (0);
}
```

13.6. TU-Dresden Matrix-Matrix Multiplication

```
/********************************************************************************
 * Fast Matrix Multiplication for Cell BE Processors
 * Copyright (C) 2007  Daniel Hackenberg, ZIH, TU-Dresden
 * A comprehensive description is available at
 * http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/
 * architektur_und_leistungsanalyse_von_hochleistungsrechnern/cell
 * Please send your feedback to daniel.hackenberg@zih.tu-dresden.de
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
 *******************************************************************************/

#include "matmul.h"
#include <libspe.h>
#include <stdio.h>
#include <sched.h>
#include <sys/wait.h>
#include <malloc_align.h>
#include <free_align.h>
#include <sys/time.h>

#define min(C,A,B) if (A<B) C=A; else C=B;

extern spe_program_handle_t matmul_spu;

control_block cb[MAX_SPE_THREADS] __attribute__ ((aligned (128)));

int num_spes = 1, size = 256, verify = 0;


void print_usage(char* text) {
  printf("USAGE: %s [options]\n", text);
  printf("       Valid Options include:\n");
  printf("         -m #   : Perform maxtrix multiply of matrices of size #.\n");
  printf("                  Has to be a mutiple of 128. Default is 256.\n");
  printf("         -s #   : Run with # number of SPEs. 1, 2, 4, 6, 8 and 16 SPEs possible.\n");
  printf("                  With a matrix of size 128 you can use 1 SPE, for matrices of size 256\n");
  printf("                  you can use 4 SPEs, and for matrices of size 512 and larger up to 16 SPEs\n");
  printf("                  are possible. The default is 1 SPE.\n");
  printf("         -v     : Verify results.\n");
  printf("         -h     : Output this usage help screen.\n");
  exit(1);
}

void print_error(char* text) {
  fprintf(stderr, "%s", text);
  exit(1);
}

double my_gettimeofday(void) {
  struct timeval time;
  gettimeofday( &time, (void *) 0);
  return (double)(time.tv_sec) + (double)time.tv_usec * 1.0e-6;
}

void verify_all(float* _matrix_A, float* _matrix_B, float* _spe_result, float* _ppe_result, int _size) {

  int i, j, k, fail = 0;
#ifdef __USE_BDL__
  int i0, j0, k0, a_offset, b_offset, c_offset;
#endif
  float delta;

  printf("Computing expected results on the PPE.\n");
  printf("This might take a while... Or even very long...\n");

#ifdef __USE_BDL__
  for (i0 = 0; i0 < _size; i0 += MATRIX_DIM) {
    for (j0 = 0; j0 < _size; j0 += MATRIX_DIM) {
      for (k0 = 0; k0 < _size; k0 += MATRIX_DIM) {
        a_offset = i0 * _size + k0 * MATRIX_DIM;
        b_offset = k0 * _size + j0 * MATRIX_DIM;
        c_offset = i0 * _size + j0 * MATRIX_DIM;
        for (i = 0; i < MATRIX_DIM; i++)
          for (j = 0; j < MATRIX_DIM; j++)
            for (k = 0; k < MATRIX_DIM; k++)
              _ppe_result[i * MATRIX_DIM + j + c_offset] += _matrix_A[i * MATRIX_DIM + k + a_offset] * _matrix_B[k * MATRIX_DIM + j + 
b_offset];
      }
    }
  }

#else
  for (i = 0; i < _size; i++)
    for (k = 0; k < _size; k++)
      for (j = 0; j < _size; j++)
        _ppe_result[i * _size + j] += _matrix_A[i * _size + k] * _matrix_B[k * _size + j];
#endif

  printf("Verifying the results... ");

  for (i = 0; i < _size; i++) {
    for (j = 0; j < _size; j++) {
      delta = _ppe_result[i * _size + j] - _spe_result[i * _size + j];
      if (delta < 0.0f) delta = -delta;
      if (delta > VERIFY_ERROR) {
        printf("  %d %d ppe=%f spe=%f\n", i, j, _ppe_result[i * _size + j], _spe_result[i* _size + j]);
        fail++;
      }
      if (fail > 50) print_error("FAILED\n");
    }
  }

  printf("PASSED\n");
  fflush(stdout);

  return;
}

void evaluate_args(int argc, char *argv[]) {

  int i;

  for (i = 1; i < argc; i++) {
    if (*argv[i] == '-') {
      switch (*(argv[i]+1)) {
      case 'm':
        i++;
        if (i < argc) {
          size = atoi(argv[i]);
          if ((size % 128) != 0) {
            fprintf(stderr, "ERROR: Invalid matrix size %d specified.\n", size);
            print_usage(argv[0]);
          }
        } else {
          fprintf(stderr, "ERROR: Specify matrix size.\n");
          print_usage(argv[0]);
        }
        break;
      case 's':
        i++;
        if (i < argc) {
          num_spes = atoi(argv[i]);
          if ((num_spes > 1) && (num_spes % 2)) {
            fprintf(stderr, "Only 1, 2, 4, 6, 8 or 16 SPEs supported.");
            print_usage(argv[0]);
          }
        } else {
          fprintf(stderr, "ERROR: Specify number of SPEs.\n");
          print_usage(argv[0]);
        }
        break;
      case 'v':
        verify = 1;
        break;
      default:
        print_usage(argv[0]);
        break;
      }
    } else print_usage(argv[0]);
  }

  if ((num_spes > 1) && (size < 256))
    print_error("ERROR: You can only use 1 SPE for matrices of size 128.\n");
  if ((num_spes > 4) && (size < 512))
    print_error("ERROR: You can only use 4 SPEs for matrices smaller than 512.\n");

  return;
}

int main(int argc, char *argv[]) {

  spe_gid_t gid;
  speid_t speid[MAX_SPE_THREADS];

  int i, status;
  int physid[MAX_SPE_THREADS];
  unsigned int spe_time[MAX_SPE_THREADS], spe_count[MAX_SPE_THREADS];
  double performance_all = 0.0, performance, t_start = 0.0, t_all = 0.0;
  float *matrix_A, *matrix_B, *spe_result, *ppe_result;

  for (i = 0; i < MAX_SPE_THREADS; i++) physid[i] = 0;

  /* parse command line arguments */
  evaluate_args(argc, argv);

  printf("\nFast matrix multiplications on Cell (SMP) systems.\n");
  printf("Copyright (C) 2007  Daniel Hackenberg, ZIH, TU-Dresden\n\n");
  printf("Running matrix multiplication of %dx%d matrices using %d SPEs...\n", size, size, num_spes);

  /* malloc call, aligned to 128 Byte */
  matrix_A   = (float *)_malloc_align(size*size*sizeof(float), 7);
  matrix_B   = (float *)_malloc_align(size*size*sizeof(float), 7);
  spe_result = (float *)_malloc_align(size*size*sizeof(float), 7);
  ppe_result = (float *)_malloc_align(size*size*sizeof(float), 7);

  printf("Initializing arrays with random numbers... ");
  fflush(stdout);

  /* initialize matrizes with random numbers */
  if (matrix_A && matrix_B && spe_result && ppe_result) {
    for (i = 0; i < size * size; i++) {
#ifdef __INIT_RAND__
      matrix_A[i]   = (RAND_MAX-(float)rand())/RAND_MAX;
      matrix_B[i]   = (RAND_MAX-(float)rand())/RAND_MAX;
      spe_result[i] = (RAND_MAX-(float)rand())/RAND_MAX;
#else
      matrix_A[i]   = 0.1;
      matrix_B[i]   = 0.2;
      spe_result[i] = 0.3;
#endif
      ppe_result[i] = spe_result[i];
    }
    printf("done!\n");
  } else print_error("Failed to allocate arrays.\n");

  /* put control block information together */
  for (i = 0; i < num_spes; i++) {
    cb[i].matrix_A   = (addr64) (unsigned long long)matrix_A;
    cb[i].matrix_B   = (addr64) (unsigned long long)matrix_B;
    cb[i].spe_result = (addr64) (unsigned long long)spe_result;
    cb[i].size       = (int) size;
    cb[i].spe_num    = (int) i;
    cb[i].num_spes   = (int) num_spes;
    cb[i].m_blocks   = (int) size / MATRIX_DIM;
    cb[i].n_blocks   = (int) size / MATRIX_DIM;
    cb[i].p_blocks   = (int) size / MATRIX_DIM;
  }

  /* Create an SPE group */
  gid = spe_create_group (SCHED_OTHER, 0, 1);
  if ((gid == NULL) || (spe_group_max (gid) < 1)) exit(1);

  /* allocate the SPE tasks */
  for (i = 0; i < num_spes; i++) {
    speid[i] = spe_create_thread (gid, &matmul_spu, (unsigned long long *) &cb[i], NULL, -1, 0);
    if (speid[i]== NULL) print_error("FAILED: spe_create_thread");
  }

  /* wait for a synchronisation signal of each SPE */
  for (i = 0; i < num_spes; i++) {
    while (!spe_stat_out_mbox(speid[i]));
    spe_read_out_mbox(speid[i]);
  }

  printf("Starting SPE calculations... ");
  fflush(stdout);

  /* start PPE-side measurement of the execution time */
  t_start = my_gettimeofday();

  /* send a start signal to each SPE */
  for (i = 0; i < num_spes; i++) spe_write_in_mbox(speid[i], 0);

  /* get the performance data of each SPE */
  for (i = 0; i < num_spes; i++) {
    while (!spe_stat_out_mbox(speid[i]));
    spe_time[i] = spe_read_out_mbox(speid[i]);
    while (!spe_stat_out_mbox(speid[i]));
    spe_count[i] = spe_read_out_mbox(speid[i]);
  }

  /* stop PPE-side measurement of execution time */
  t_all = my_gettimeofday() - t_start;

  /* wait until all SPE threads have finished */
  for (i = 0; i < num_spes; i++) spe_wait(speid[i], &status, 0);

  if (!WIFEXITED(status)) print_error("FAILED: SPE abnormally terminated\n");
  else printf("done!\n");

  /* start result verification */
  if (verify) verify_all(matrix_A, matrix_B, spe_result, ppe_result, size);

  /* print performance results */
  printf("\nPerformance results assuming a clock frequency of %d MHz and a timebase of %d:\n", __freq__, __timebase__);

  for(i = 0; i < num_spes; i++) {
    performance = ((double)spe_count[i]*2.0*MATRIX_DIM*MATRIX_DIM*MATRIX_DIM*2.4)/(((double)spe_time[i]*2400000000.0/__timebase__));
    printf("Performance of SPE %2d: %.2lf GFLOPS\n", i, performance);
    performance_all += performance;
  }

  if (num_spes > 1)
    printf("Aggregated performance for all %d SPEs: %.2lf GFLOPS (of %.2f GFLOPS theoretical peak).\n", num_spes,  performance_all,
__freq__*8.0*(float)num_spes/1000.0);

  printf("\nPPE-measured performance of matrix multiplication using %d SPEs: %.2lf GFLOPS.\n", num_spes,
(2.0*(double)size*(double)size*(double)size*1.0e-9)/t_all);
  printf("(PPE-side measurement is inaccurate for small matrices!).\n\n");

  fflush(stdout);

  _free_align(matrix_A);
  _free_align(matrix_B);
  _free_align(spe_result);
  return 0;
}


/*******************************************************************************
 * Fast Matrix Multiplication for Cell BE Processors
 * Copyright (C) 2007  Daniel Hackenberg, ZIH, TU-Dresden
 * A comprehensive description is available at
 * http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/
 * architektur_und_leistungsanalyse_von_hochleistungsrechnern/cell
 * Please send your feedback to daniel.hackenberg@zih.tu-dresden.de
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
 ******************************************************************************/

#include "matmul.h"
#include <spu_mfcio.h>

#define __dma__(VAR1, VAR2) { \
  vlist[VAR1] = v0;           \
  vlist[VAR2] = v1;           \
  v0 = spu_add(v0, add);      \
  v1 = spu_add(v1, add);      \
}

control_block cb __attribute__ ((aligned (128)));

unsigned int dma_list1a[128];
unsigned int dma_list2a[128];
unsigned int dma_list3a[128];

unsigned int dma_list1b[128];
unsigned int dma_list2b[128];
unsigned int dma_list3b[128];

unsigned int dma_list3c[128];
unsigned int dma_list3d[128];

unsigned int dma_list4a[128];
unsigned int dma_list4b[128];
unsigned int dma_list4c[128];
unsigned int dma_list4d[128];

addr64 PPE_matrix_A_Ptr, PPE_matrix_B_Ptr, PPE_matrix_C_Ptr;

unsigned int size;
int m_blocks, n_blocks, p_blocks;

/* This vector is used by the assembly-coded matmul function. */
vector unsigned int SIMDadd =  (vector unsigned int){0, 64, 128, 192};

/* This is the assembly-coded matmul function. */
extern void matmul_SIMD64(float *_matrix_A, float *_matrix_B, float *_result);

/*
 * This is a fast and completely unrolled function that generates the DMA lists
 * that will be used to load or store the block matrices.
 */
void fill_dma_list(unsigned int _base, unsigned int *_list_addr, int _offset_m, int _offset_n) {
  unsigned int k;
  vector unsigned int v0, v1, add;
  vector unsigned int *vlist;

  add =  (vector unsigned int){0, size * 16, 0, size * 16};
  vlist = (vector unsigned int *)_list_addr;

  k = _base + (_offset_m * DMA_SIZE * n_blocks * MATRIX_DIM) + (_offset_n * DMA_SIZE);

  v0 = (vector unsigned int){DMA_SIZE, k, DMA_SIZE, k + size*4};
  v1 = (vector unsigned int){DMA_SIZE, k + size * 8, DMA_SIZE, k + size * 12};

  __dma__(0,1);
  __dma__(2,3);
  __dma__(4,5);
  __dma__(6,7);
  __dma__(8,9);
  __dma__(10,11);
  __dma__(12,13);
  __dma__(14,15);
  __dma__(16,17);
  __dma__(18,19);
  __dma__(20,21);
  __dma__(22,23);
  __dma__(24,25);
  __dma__(26,27);
  __dma__(28,29);
  __dma__(30,31);

  return;
}
```

```
inline void wait_for_mbox(int _tag_id) {
  mfc_write_tag_mask(1 << _tag_id);
  mfc_read_tag_status_all();
  return;
}

inline void load_data(addr64 _PPE_Ptr, float *_feld, unsigned int *_dma_list, int _offset_m, int _offset_n, int _tag) {
#ifdef __USE_BDL__
  mfc_get(_feld, _PPE_Ptr.ull + (unsigned long long)((_offset_m * n_blocks * MATRIX_SIZE) + (_offset_n * MATRIX_SIZE)), MATRIX_SIZE,
_tag, 0, 0);
#else
  fill_dma_list(_PPE_Ptr.ui[1], _dma_list, _offset_m, _offset_n);
  mfc_getl(_feld, _PPE_Ptr.ull, _dma_list, DMA_LIST_SIZE, _tag, 0, 0);
#endif
}

inline void store_data(float *_feld, unsigned int *_dma_list, int _offset_m, int _offset_n, int _tag) {
#ifdef __USE_BDL__
  mfc_put(_feld, PPE_matrix_C_Ptr.ull + (unsigned long long)((_offset_m * n_blocks * MATRIX_SIZE) + (_offset_n * MATRIX_SIZE)),
MATRIX_SIZE, _tag, 0, 0);
#else
  fill_dma_list(PPE_matrix_C_Ptr.ui[1], _dma_list, _offset_m, _offset_n);
  mfc_putl(_feld, PPE_matrix_C_Ptr.ull, _dma_list, DMA_LIST_SIZE, _tag, 0, 0);
#endif
}

int main(unsigned long long speid, addr64 argp, addr64 envp) {

  int m, n, p = 0;
  int offset_1_a_m, offset_1_b_m, offset_2_a_m, offset_2_b_m, offset_3_a_m, offset_3_b_m;
  int offset_1_a_n, offset_1_b_n, offset_2_a_n, offset_2_b_n, offset_3_a_n, offset_3_b_n;
  int offset_3_c_m, offset_3_d_m, offset_3_c_n, offset_3_d_n;
  unsigned int t_start = 0, t_spu, count = 0;


/* Multibuffering is used to do the calculations efficiently.
 * There are two buffers for matrix A, two buffers for matrix B
 * and four buffers for matrix C.
 * Capital letters (A, B, C) refer to the matrix itself.
 * Lower case letters (a, b, c, d) refer to the buffers in the local store.
 *
 *          B
 *    ---------
 *   |   | a b |
 *   |---------|
 * A | a | a b | C
 *   | b | c d |
 *    ---------
 */
  float matrix_A_a[MATRIX_ENTRIES] __attribute__ ((aligned (128)));
  float matrix_A_b[MATRIX_ENTRIES] __attribute__ ((aligned (128)));

  float matrix_B_a[MATRIX_ENTRIES] __attribute__ ((aligned (128)));
  float matrix_B_b[MATRIX_ENTRIES] __attribute__ ((aligned (128)));

  float matrix_C_a[MATRIX_ENTRIES] __attribute__ ((aligned (128)));
  float matrix_C_b[MATRIX_ENTRIES] __attribute__ ((aligned (128)));
  float matrix_C_c[MATRIX_ENTRIES] __attribute__ ((aligned (128)));
  float matrix_C_d[MATRIX_ENTRIES] __attribute__ ((aligned (128)));

  /* tell the PPE that the SPE thread is running */
  spu_write_out_mbox(0);

  /* this mailbox call blocks until the PPE has created the control block */
  spu_read_in_mbox();

  /* start the SPE-side measurement of the execution time */
  spu_write_decrementer(0x7fffffff);
  t_start = spu_read_decrementer();

  /* fetch the control block via DMA */
  mfc_get(&cb, argp.ui[1], sizeof(cb), 1, 0, 0);
  mfc_write_tag_mask(1<<1);
  mfc_read_tag_status_all();

  /* set local variables and  pointers according to the control block information */
  PPE_matrix_A_Ptr = cb.matrix_A;
  PPE_matrix_B_Ptr = cb.matrix_B;
  PPE_matrix_C_Ptr = cb.spe_result;

  size = cb.size;
  m_blocks = cb.m_blocks;
  n_blocks = cb.n_blocks;
  p_blocks = cb.p_blocks;

/*
 *         n columns
 *         ----->
 *         |
 * m rows  |
 *         V
 */

  m = 2 * (cb.spe_num / (n_blocks / 2));
  n = 2 * (cb.spe_num % (n_blocks / 2));

  offset_1_a_m = m;
  offset_1_a_n = 0;
  offset_1_b_m = m + 1;
  offset_1_b_n = 0;
  offset_2_a_m = 0;
  offset_2_a_n = n;
  offset_2_b_m = 0;
  offset_2_b_n = n + 1;
  offset_3_a_m = m;
  offset_3_a_n = n;
  offset_3_b_m = m;
  offset_3_b_n = n + 1;
  offset_3_c_m = m + 1;
  offset_3_c_n = n;
  offset_3_d_m = m + 1;
  offset_3_d_n = n + 1;

/*
 *          B
 *    ---------
 *   |   | a b |
 *   |---------|
 * A | a | a b | C
 *   | b | c d |
 *    ---------
 */

  load_data(PPE_matrix_A_Ptr, matrix_A_a, dma_list1a, offset_1_a_m, offset_1_a_n, 1);
  load_data(PPE_matrix_B_Ptr, matrix_B_a, dma_list2a, offset_2_a_m, offset_2_a_n, 1);
  load_data(PPE_matrix_C_Ptr, matrix_C_a, dma_list3a, offset_3_a_m, offset_3_a_n, 1);

  load_data(PPE_matrix_B_Ptr, matrix_B_b, dma_list2b, offset_2_b_m, offset_2_b_n, 2);
  load_data(PPE_matrix_C_Ptr, matrix_C_b, dma_list3b, offset_3_b_m, offset_3_b_n, 2);


  while (m < m_blocks) {
    n += cb.num_spes * 2;

    load_data(PPE_matrix_C_Ptr, matrix_C_c, dma_list3c, offset_3_c_m, offset_3_c_n, 3);
    load_data(PPE_matrix_C_Ptr, matrix_C_d, dma_list3d, offset_3_d_m, offset_3_d_n, 3);

    for (p = 1; p < p_blocks; p++) {
      wait_for_mbox(1);
      matmul_SIMD64(matrix_A_a, matrix_B_a, matrix_C_a);           /* mul a a a */
      count++;

      load_data(PPE_matrix_A_Ptr, matrix_A_b, dma_list1b, offset_1_b_m, offset_1_b_n + p - 1, 3);

      wait_for_mbox(2);
      matmul_SIMD64(matrix_A_a, matrix_B_b, matrix_C_b);           /* mul a b b */
      count++;

      load_data(PPE_matrix_A_Ptr, matrix_A_a, dma_list1a, offset_1_a_m, offset_1_a_n + p, 1);

      wait_for_mbox(3);
      matmul_SIMD64(matrix_A_b, matrix_B_a, matrix_C_c);           /* mul b a c */
      count++;

      load_data(PPE_matrix_B_Ptr, matrix_B_a, dma_list2a, offset_2_a_m + p, offset_2_a_n, 1);

      matmul_SIMD64(matrix_A_b, matrix_B_b, matrix_C_d);           /* mul b b d */
      count++;

      load_data(PPE_matrix_B_Ptr, matrix_B_b, dma_list2b, offset_2_b_m + p, offset_2_b_n, 2);
    }

    while (n >= n_blocks) {
      m += 2;
      n -= n_blocks;
    }

    if (m < m_blocks) {

      wait_for_mbox(1);
      matmul_SIMD64(matrix_A_a, matrix_B_a, matrix_C_a);           /* mul a a a */
      count++;

      load_data(PPE_matrix_A_Ptr, matrix_A_b, dma_list1b, offset_1_b_m, offset_1_b_n + cb.p_blocks - 1, 3);
      store_data(matrix_C_a, dma_list4a, offset_3_a_m, offset_3_a_n, 4);
      offset_3_a_m = m;
      offset_3_a_n = n;

      offset_1_a_m = m;
      offset_1_a_n = 0;
      offset_1_b_m = m + 1;
      offset_1_b_n = 0;
      offset_2_a_m = 0;
      offset_2_a_n = n;
      offset_2_b_m = 0;
      offset_2_b_n = n + 1;

      wait_for_mbox(2);
      matmul_SIMD64(matrix_A_a, matrix_B_b, matrix_C_b);           /* mul a b b */
      count++;

      load_data(PPE_matrix_A_Ptr, matrix_A_a, dma_list1a, offset_1_a_m, offset_1_a_n, 1);
      store_data(matrix_C_b, dma_list4b, offset_3_b_m, offset_3_b_n, 5);
      offset_3_b_m = m;
      offset_3_b_n = n + 1;

      wait_for_mbox(3);
      matmul_SIMD64(matrix_A_b, matrix_B_a, matrix_C_c);           /* mul b a c */
      count++;

      load_data(PPE_matrix_B_Ptr, matrix_B_a, dma_list2a, offset_2_a_m, offset_2_a_n, 1);
      store_data(matrix_C_c, dma_list4c, offset_3_c_m, offset_3_c_n, 6);
      offset_3_c_m = m + 1;
      offset_3_c_n = n;
      wait_for_mbox(4);
      load_data(PPE_matrix_C_Ptr, matrix_C_a, dma_list3a, offset_3_a_m, offset_3_a_n, 1);

      matmul_SIMD64(matrix_A_b, matrix_B_b, matrix_C_d);           /* mul b b d */
      count++;

      load_data(PPE_matrix_B_Ptr, matrix_B_b, dma_list2b, offset_2_b_m, offset_2_b_n, 2);
      store_data(matrix_C_d, dma_list4d, offset_3_d_m, offset_3_d_n, 6);
      offset_3_d_m = m + 1;
      offset_3_d_n = n + 1;

      wait_for_mbox(5);
      load_data(PPE_matrix_C_Ptr, matrix_C_b, dma_list3b, offset_3_b_m, offset_3_b_n, 2);

      wait_for_mbox(6);

    } else {

      wait_for_mbox(1);
      matmul_SIMD64(matrix_A_a, matrix_B_a, matrix_C_a);           /* mul a a a */
      count++;

      load_data(PPE_matrix_A_Ptr, matrix_A_b, dma_list1b, offset_1_b_m, offset_1_b_n + cb.p_blocks - 1, 3);
      store_data(matrix_C_a, dma_list4a, offset_3_a_m, offset_3_a_n, 4);

      wait_for_mbox(2);
      matmul_SIMD64(matrix_A_a, matrix_B_b, matrix_C_b);           /* mul a b b */
      count++;

      store_data(matrix_C_b, dma_list4b, offset_3_b_m, offset_3_b_n, 4);

      wait_for_mbox(3);
      matmul_SIMD64(matrix_A_b, matrix_B_a, matrix_C_c);           /* mul b a c */
      count++;

      store_data(matrix_C_c, dma_list4c, offset_3_c_m, offset_3_c_n, 4);

      matmul_SIMD64(matrix_A_b, matrix_B_b, matrix_C_d);           /* mul b b d */
      count++;

      store_data(matrix_C_d, dma_list4d, offset_3_d_m, offset_3_d_n, 4);

      wait_for_mbox(4);

    }

  }

  /* stop the SPE-side measurement of the execution time */
  t_spu = t_start - spu_read_decrementer();

  spu_write_out_mbox(t_spu);
  spu_write_out_mbox(count);

  return 0;
}


/********************************************************************************
 * Fast Matrix Multiplication for Cell BE Processors
 * Copyright (C) 2007  Daniel Hackenberg, ZIH, TU-Dresden
 * A comprehensive description is available at
 * http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/
 * architektur_und_leistungsanalyse_von_hochleistungsrechnern/cell
 * Please send your feedback to daniel.hackenberg@zih.tu-dresden.de
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
 ********************************************************************************/

#include <stdlib.h>

/* Define your CPU frequency in MHz. */
#define __freq__ 3200

/* Define your timebase. */
//#define __timebase__ 14318000

/* for the PS3 */
#define __timebase__ 79800000

/* Comment this if you want to use row major layout. */
#define __USE_BDL__

/* Random initialization takes a lot of time.
 * Comment this to use fast static initialization.
 */
#define __INIT_RAND__

/* For large matrices the verification might fail due to
 * the limited accuracy of single precision floats.
 * Adjust this value accordingly.
 */
#define VERIFY_ERROR 0.1

#define MAX_SPE_THREADS 16

/* You really shouldn't change anything below. */
#define MATRIX_DIM           64
#define MATRIX_ENTRIES       MATRIX_DIM * MATRIX_DIM
#define MATRIX_SIZE          MATRIX_ENTRIES * sizeof(float)
#define DMA_SIZE             MATRIX_DIM * sizeof(float)
#define DMA_LIST_SIZE        MATRIX_DIM * 2 * sizeof(float)


typedef union {
  unsigned long long ull;
  unsigned int ui[2];
} addr64;

typedef struct _control_block {
  addr64 matrix_A;
  addr64 matrix_B;
  addr64 spe_result;
  int size;
  int spe_num;
  int num_spes;
  int m_blocks;
  int n_blocks;
  int p_blocks;
  unsigned char pad[74];   /* pad to 128 bytes */
} control_block;
```

```
# Extremely fast 64x64 matmul funktion for Cell BE Processors
# Please don't expect this assembly code to be documented.
# Copyright (C) 2007  Daniel Hackenberg, ZIH, TU-Dresden
# Please send your feedback to daniel.hackenberg@zih.tu-dresden.de
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA

          .file     "matmul_spu_simd.c"
.text
          .align    3
          .global   matmul_SIMD64
          .type     matmul_SIMD64, @function
matmul_SIMD64:
          ori       $9,$3,0
          rotqbyi   $10,$4,0
          il        $54,16
          rotqbyi   $11,$5,0


          ori       $59,$3,0
          lqd       $16,0($9)
          ori       $60,$5,0
          lqd       $17,256($9)
          ila       $12,66051
          lqd       $18,512($9)
          ilhu      $13,1029
          lqd       $19,768($9)
          iohl      $13,1543
          lqd       $21,0($10)
          ilhu      $14,2057
          lqd       $22,16($10)
          iohl      $14,2571
          lqd       $23,32($10)
          ilhu      $15,3085
          lqd       $24,48($10)
          iohl      $15,3599
          shufb     $29,$16,$16,$12
          il        $56,1024
          shufb     $30,$17,$17,$12
          ai        $77,$11,64
          shufb     $31,$18,$18,$12
          il        $78,8192
          shufb     $32,$19,$19,$12

          lqa       $79,SIMDadd
          shufb     $55,$10,$10,$12
          lqd       $37,0($11)
          lqd       $38,16($11)
          lqd       $39,32($11)
          lqd       $40,48($11)
          lqd       $41,256($11)
          lqd       $42,272($11)
          a         $55,$55,$79

          lqd       $71,544($77)
          lqd       $72,560($77)
          lqd       $73,768($77)
          lqd       $74,784($77)
          lqd       $75,800($77)
```

```
         a         $58,$55,$78
         lqd       $76,816($77)

.LOOP:

#N00-N15

#1
         fma       $37,$29,$21,$37
         lqd       $43,288($11)
         fma       $38,$29,$22,$38
         lqd       $44,304($11)
         fma       $39,$29,$23,$39
         lqd       $45,512($11)
         fma       $40,$29,$24,$40
         lqd       $46,528($11)

         fma       $41,$30,$21,$41
         lqd       $47,544($11)
         fma       $42,$30,$22,$42
         lqd       $48,560($11)
         fma       $43,$30,$23,$43
         lqd       $49,768($11)
         fma       $44,$30,$24,$44
         lqd       $50,784($11)

         fma       $45,$31,$21,$45
         lqd       $51,800($11)
         fma       $46,$31,$22,$46
         lqd       $52,816($11)
         fma       $47,$31,$23,$47
         lqd       $25,256($10)
         fma       $48,$31,$24,$48
         lqd       $26,272($10)

         fma       $49,$32,$21,$49
         lqd       $27,288($10)
         fma       $50,$32,$22,$50
         shufb     $33,$16,$16,$13
         fma       $51,$32,$23,$51
         lqd       $28,304($10)
         fma       $52,$32,$24,$52
         shufb     $34,$17,$17,$13

         ai        $54,$54,-1
         lnop
         fma       $37,$33,$25,$37
         fma       $38,$33,$26,$38
         fma       $39,$33,$27,$39
         shufb     $35,$18,$18,$13
         fma       $40,$33,$28,$40
         shufb     $36,$19,$19,$13

         fma       $41,$34,$25,$41
         lqd       $21,512($10)
         fma       $42,$34,$26,$42
         lqd       $22,528($10)
         fma       $43,$34,$27,$43
         lqd       $23,544($10)
         fma       $44,$34,$28,$44
         lqd       $24,560($10)

         fma       $45,$35,$25,$45
         shufb     $29,$16,$16,$14
         fma       $46,$35,$26,$46
         shufb     $30,$17,$17,$14
         fma       $47,$35,$27,$47
         shufb     $31,$18,$18,$14
         fma       $48,$35,$28,$48
         shufb     $32,$19,$19,$14

         fma       $49,$36,$25,$49
         fma       $50,$36,$26,$50
         fma       $51,$36,$27,$51
         fma       $52,$36,$28,$52


         fma       $37,$29,$21,$37
         lqd       $25,768($10)
         fma       $38,$29,$22,$38
         lqd       $26,784($10)
         fma       $39,$29,$23,$39
         lqd       $27,800($10)
         fma       $40,$29,$24,$40
         lqd       $28,816($10)

         fma       $41,$30,$21,$41
         shufb     $33,$16,$16,$15
         fma       $42,$30,$22,$42
         shufb     $34,$17,$17,$15
         fma       $43,$30,$23,$43
         shufb     $35,$18,$18,$15
         fma       $44,$30,$24,$44
         shufb     $36,$19,$19,$15

         fma       $45,$31,$21,$45
         fma       $46,$31,$22,$46
         fma       $47,$31,$23,$47
         fma       $48,$31,$24,$48

         fma       $49,$32,$21,$49
         fma       $50,$32,$22,$50
         fma       $51,$32,$23,$51
```

```
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          lqd       $16,16($9)
          fma       $38,$33,$26,$38
          lqd       $17,272($9)
          fma       $39,$33,$27,$39
          lqd       $18,528($9)
          fma       $40,$33,$28,$40
          lqd       $19,784($9)

          fma       $41,$34,$25,$41
          lqd       $21,1024($10)
          fma       $42,$34,$26,$42
          lqd       $22,1040($10)
          fma       $43,$34,$27,$43
          lqd       $23,1056($10)
          fma       $44,$34,$28,$44
          lqd       $24,1072($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          stqd      $71,544($77)
          fma       $50,$36,$26,$50
          stqd      $72,560($77)
          fma       $51,$36,$27,$51
          stqd      $73,768($77)
          fma       $52,$36,$28,$52
          stqd      $74,784($77)



#2
          fma       $37,$29,$21,$37
          stqd      $75,800($77)
          fma       $38,$29,$22,$38
          stqd      $76,816($77)
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40
          ai        $77,$11,64
          lnop

          fma       $41,$30,$21,$41
          lqd       $25,1280($10)
          fma       $42,$30,$22,$42
          lqd       $26,1296($10)
          fma       $43,$30,$23,$43
          lqd       $27,1312($10)
          fma       $44,$30,$24,$44
          lqd       $28,1328($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          lqd       $61,0($77)
          fma       $50,$32,$22,$50
          lqd       $62,16($77)
          fma       $51,$32,$23,$51
          lqd       $63,32($77)
          fma       $52,$32,$24,$52
          lqd       $64,48($77)

          fma       $37,$33,$25,$37
          lqd       $65,256($77)
          fma       $38,$33,$26,$38
          lqd       $66,272($77)
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,1536($10)
          fma       $42,$34,$26,$42
          lqd       $22,1552($10)
          fma       $43,$34,$27,$43
          lqd       $23,1568($10)
          fma       $44,$34,$28,$44
          lqd       $24,1584($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14
          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,1792($10)
          fma      $38,$29,$22,$38
          lqd      $26,1808($10)
          fma      $39,$29,$23,$39
          lqd      $27,1824($10)
          fma      $40,$29,$24,$40
          lqd      $28,1840($10)

          fma      $41,$30,$21,$41
          shufb    $33,$16,$16,$15
          fma      $42,$30,$22,$42
          shufb    $34,$17,$17,$15
          fma      $43,$30,$23,$43
          shufb    $35,$18,$18,$15
          fma      $44,$30,$24,$44
          shufb    $36,$19,$19,$15

          fma      $45,$31,$21,$45
          fma      $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,32($9)
          fma      $38,$33,$26,$38
          lqd      $17,288($9)
          fma      $39,$33,$27,$39
          lqd      $18,544($9)
          fma      $40,$33,$28,$40
          lqd      $19,800($9)

          fma      $41,$34,$25,$41
          lqd      $21,2048($10)
          fma      $42,$34,$26,$42
          lqd      $22,2064($10)
          fma      $43,$34,$27,$43
          lqd      $23,2080($10)
          fma      $44,$34,$28,$44
          lqd      $24,2096($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$12
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$12
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$12
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$12

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#3
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,2304($10)
          fma      $42,$30,$22,$42
          lqd      $26,2320($10)
          fma      $43,$30,$23,$43
          lqd      $27,2336($10)
          fma      $44,$30,$24,$44
          lqd      $28,2352($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40
          fma       $41,$34,$25,$41
          lqd       $21,2560($10)
          fma       $42,$34,$26,$42
          lqd       $22,2576($10)
          fma       $43,$34,$27,$43
          lqd       $23,2592($10)
          fma       $44,$34,$28,$44
          lqd       $24,2608($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,2816($10)
          fma       $38,$29,$22,$38
          lqd       $26,2832($10)
          fma       $39,$29,$23,$39
          lqd       $27,2848($10)
          fma       $40,$29,$24,$40
          lqd       $28,2864($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,48($9)
          fma       $38,$33,$26,$38
          lqd       $17,304($9)
          fma       $39,$33,$27,$39
          lqd       $18,560($9)
          fma       $40,$33,$28,$40
          lqd       $19,816($9)

          fma       $41,$34,$25,$41
          lqd       $21,3072($10)
          fma       $42,$34,$26,$42
          lqd       $22,3088($10)
          fma       $43,$34,$27,$43
          lqd       $23,3104($10)
          fma       $44,$34,$28,$44
          lqd       $24,3120($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#4
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,3328($10)
          fma       $42,$30,$22,$42
          lqd       $26,3344($10)
          fma       $43,$30,$23,$43
          lqd       $27,3360($10)
          fma       $44,$30,$24,$44
          lqd       $28,3376($10)

          fma       $45,$31,$21,$45
```

```
shufb    $33,$16,$16,$13
fma      $46,$31,$22,$46
shufb    $34,$17,$17,$13
fma      $47,$31,$23,$47
shufb    $35,$18,$18,$13
fma      $48,$31,$24,$48
shufb    $36,$19,$19,$13

fma      $49,$32,$21,$49
fma      $50,$32,$22,$50
fma      $51,$32,$23,$51
fma      $52,$32,$24,$52

fma      $37,$33,$25,$37
fma      $38,$33,$26,$38
fma      $39,$33,$27,$39
fma      $40,$33,$28,$40

fma      $41,$34,$25,$41
lqd      $21,3584($10)
fma      $42,$34,$26,$42
lqd      $22,3600($10)
fma      $43,$34,$27,$43
lqd      $23,3616($10)
fma      $44,$34,$28,$44
lqd      $24,3632($10)

fma      $45,$35,$25,$45
shufb    $29,$16,$16,$14
fma      $46,$35,$26,$46
shufb    $30,$17,$17,$14
fma      $47,$35,$27,$47
shufb    $31,$18,$18,$14
fma      $48,$35,$28,$48
shufb    $32,$19,$19,$14

fma      $49,$36,$25,$49
fma      $50,$36,$26,$50
fma      $51,$36,$27,$51
fma      $52,$36,$28,$52


fma      $37,$29,$21,$37
lqd      $25,3840($10)
fma      $38,$29,$22,$38
lqd      $26,3856($10)
fma      $39,$29,$23,$39
lqd      $27,3872($10)
fma      $40,$29,$24,$40
lqd      $28,3888($10)

fma      $41,$30,$21,$41
shufb    $33,$16,$16,$15
fma      $42,$30,$22,$42
shufb    $34,$17,$17,$15
fma      $43,$30,$23,$43
shufb    $35,$18,$18,$15
fma      $44,$30,$24,$44
shufb    $36,$19,$19,$15

fma      $45,$31,$21,$45
fma      $46,$31,$22,$46
fma      $47,$31,$23,$47
fma      $48,$31,$24,$48

fma      $49,$32,$21,$49
fma      $50,$32,$22,$50
fma      $51,$32,$23,$51
fma      $52,$32,$24,$52


fma      $37,$33,$25,$37
lqd      $16,64($9)
fma      $38,$33,$26,$38
lqd      $17,320($9)
fma      $39,$33,$27,$39
lqd      $18,576($9)
fma      $40,$33,$28,$40
lqd      $19,832($9)

fma      $41,$34,$25,$41
lqd      $21,4096($10)
fma      $42,$34,$26,$42
lqd      $22,4112($10)
fma      $43,$34,$27,$43
lqd      $23,4128($10)
fma      $44,$34,$28,$44
lqd      $24,4144($10)

fma      $45,$35,$25,$45
shufb    $29,$16,$16,$12
fma      $46,$35,$26,$46
shufb    $30,$17,$17,$12
fma      $47,$35,$27,$47
shufb    $31,$18,$18,$12
fma      $48,$35,$28,$48
shufb    $32,$19,$19,$12

fma      $49,$36,$25,$49
fma      $50,$36,$26,$50
fma      $51,$36,$27,$51
fma      $52,$36,$28,$52
```

```
#5
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,4352($10)
          fma      $42,$30,$22,$42
          lqd      $26,4368($10)
          fma      $43,$30,$23,$43
          lqd      $27,4384($10)
          fma      $44,$30,$24,$44
          lqd      $28,4400($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,4608($10)
          fma      $42,$34,$26,$42
          lqd      $22,4624($10)
          fma      $43,$34,$27,$43
          lqd      $23,4640($10)
          fma      $44,$34,$28,$44
          lqd      $24,4656($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,4864($10)
          fma      $38,$29,$22,$38
          lqd      $26,4880($10)
          fma      $39,$29,$23,$39
          lqd      $27,4896($10)
          fma      $40,$29,$24,$40
          lqd      $28,4912($10)

          fma      $41,$30,$21,$41
          shufb    $33,$16,$16,$15
          fma      $42,$30,$22,$42
          shufb    $34,$17,$17,$15
          fma      $43,$30,$23,$43
          shufb    $35,$18,$18,$15
          fma      $44,$30,$24,$44
          shufb    $36,$19,$19,$15

          fma      $45,$31,$21,$45
          fma      $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,80($9)
          fma      $38,$33,$26,$38
          lqd      $17,336($9)
          fma      $39,$33,$27,$39
          lqd      $18,592($9)
          fma      $40,$33,$28,$40
          lqd      $19,848($9)

          fma      $41,$34,$25,$41
          lqd      $21,5120($10)
          fma      $42,$34,$26,$42
          lqd      $22,5136($10)
          fma      $43,$34,$27,$43
```

```
        lqd     $23,5152($10)
        fma     $44,$34,$28,$44
        lqd     $24,5168($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$12
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$12
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$12
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$12

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52



#6
        fma     $37,$29,$21,$37
        fma     $38,$29,$22,$38
        fma     $39,$29,$23,$39
        fma     $40,$29,$24,$40

        fma     $41,$30,$21,$41
        lqd     $25,5376($10)
        fma     $42,$30,$22,$42
        lqd     $26,5392($10)
        fma     $43,$30,$23,$43
        lqd     $27,5408($10)
        fma     $44,$30,$24,$44
        lqd     $28,5424($10)

        fma     $45,$31,$21,$45
        shufb   $33,$16,$16,$13
        fma     $46,$31,$22,$46
        shufb   $34,$17,$17,$13
        fma     $47,$31,$23,$47
        shufb   $35,$18,$18,$13
        fma     $48,$31,$24,$48
        shufb   $36,$19,$19,$13

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52

        fma     $37,$33,$25,$37
        fma     $38,$33,$26,$38
        fma     $39,$33,$27,$39
        fma     $40,$33,$28,$40

        fma     $41,$34,$25,$41
        lqd     $21,5632($10)
        fma     $42,$34,$26,$42
        lqd     $22,5648($10)
        fma     $43,$34,$27,$43
        lqd     $23,5664($10)
        fma     $44,$34,$28,$44
        lqd     $24,5680($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$14
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$14
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$14
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$14

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52


        fma     $37,$29,$21,$37
        lqd     $25,5888($10)
        fma     $38,$29,$22,$38
        lqd     $26,5904($10)
        fma     $39,$29,$23,$39
        lqd     $27,5920($10)
        fma     $40,$29,$24,$40
        lqd     $28,5936($10)

        fma     $41,$30,$21,$41
        shufb   $33,$16,$16,$15
        fma     $42,$30,$22,$42
        shufb   $34,$17,$17,$15
        fma     $43,$30,$23,$43
        shufb   $35,$18,$18,$15
        fma     $44,$30,$24,$44
        shufb   $36,$19,$19,$15

        fma     $45,$31,$21,$45
        fma     $46,$31,$22,$46
        fma     $47,$31,$23,$47
        fma     $48,$31,$24,$48

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
```

```
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,96($9)
          fma      $38,$33,$26,$38
          lqd      $17,352($9)
          fma      $39,$33,$27,$39
          lqd      $18,608($9)
          fma      $40,$33,$28,$40
          lqd      $19,864($9)

          fma      $41,$34,$25,$41
          lqd      $21,6144($10)
          fma      $42,$34,$26,$42
          lqd      $22,6160($10)
          fma      $43,$34,$27,$43
          lqd      $23,6176($10)
          fma      $44,$34,$28,$44
          lqd      $24,6192($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$12
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$12
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$12
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$12

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#7
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,6400($10)
          fma      $42,$30,$22,$42
          lqd      $26,6416($10)
          fma      $43,$30,$23,$43
          lqd      $27,6432($10)
          fma      $44,$30,$24,$44
          lqd      $28,6448($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,6656($10)
          fma      $42,$34,$26,$42
          lqd      $22,6672($10)
          fma      $43,$34,$27,$43
          lqd      $23,6688($10)
          fma      $44,$34,$28,$44
          lqd      $24,6704($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,6912($10)
          fma      $38,$29,$22,$38
          lqd      $26,6928($10)
          fma      $39,$29,$23,$39
          lqd      $27,6944($10)
          fma      $40,$29,$24,$40
```

```
        lqd     $28,6960($10)

        fma     $41,$30,$21,$41
        shufb   $33,$16,$16,$15
        fma     $42,$30,$22,$42
        shufb   $34,$17,$17,$15
        fma     $43,$30,$23,$43
        shufb   $35,$18,$18,$15
        fma     $44,$30,$24,$44
        shufb   $36,$19,$19,$15

        fma     $45,$31,$21,$45
        fma     $46,$31,$22,$46
        fma     $47,$31,$23,$47
        fma     $48,$31,$24,$48

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52


        fma     $37,$33,$25,$37
        lqd     $16,112($9)
        fma     $38,$33,$26,$38
        lqd     $17,368($9)
        fma     $39,$33,$27,$39
        lqd     $18,624($9)
        fma     $40,$33,$28,$40
        lqd     $19,880($9)

        fma     $41,$34,$25,$41
        lqd     $21,7168($10)
        fma     $42,$34,$26,$42
        lqd     $22,7184($10)
        fma     $43,$34,$27,$43
        lqd     $23,7200($10)
        fma     $44,$34,$28,$44
        lqd     $24,7216($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$12
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$12
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$12
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$12

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52



#8
        fma     $37,$29,$21,$37
        fma     $38,$29,$22,$38
        fma     $39,$29,$23,$39
        fma     $40,$29,$24,$40

        fma     $41,$30,$21,$41
        lqd     $25,7424($10)
        fma     $42,$30,$22,$42
        lqd     $26,7440($10)
        fma     $43,$30,$23,$43
        lqd     $27,7456($10)
        fma     $44,$30,$24,$44
        lqd     $28,7472($10)

        fma     $45,$31,$21,$45
        shufb   $33,$16,$16,$13
        fma     $46,$31,$22,$46
        shufb   $34,$17,$17,$13
        fma     $47,$31,$23,$47
        shufb   $35,$18,$18,$13
        fma     $48,$31,$24,$48
        shufb   $36,$19,$19,$13

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52

        fma     $37,$33,$25,$37
        fma     $38,$33,$26,$38
        fma     $39,$33,$27,$39
        fma     $40,$33,$28,$40

        fma     $41,$34,$25,$41
        lqd     $21,7680($10)
        fma     $42,$34,$26,$42
        lqd     $22,7696($10)
        fma     $43,$34,$27,$43
        lqd     $23,7712($10)
        fma     $44,$34,$28,$44
        lqd     $24,7728($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$14
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$14
```

```
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,7936($10)
          fma       $38,$29,$22,$38
          lqd       $26,7952($10)
          fma       $39,$29,$23,$39
          lqd       $27,7968($10)
          fma       $40,$29,$24,$40
          lqd       $28,7984($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
#         a         $10,$10,$78
          shufb     $10,$58,$58,$12
          fma       $48,$31,$24,$48
          lnop

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,128($9)
          fma       $38,$33,$26,$38
          lqd       $17,384($9)
          fma       $39,$33,$27,$39
          lqd       $18,640($9)
          fma       $40,$33,$28,$40
          lqd       $19,896($9)

          fma       $41,$34,$25,$41
          lqd       $21,0($10)
          fma       $42,$34,$26,$42
          lqd       $22,16($10)
          fma       $43,$34,$27,$43
          lqd       $23,32($10)
          fma       $44,$34,$28,$44
          lqd       $24,48($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#9
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,256($10)
          fma       $42,$30,$22,$42
          lqd       $26,272($10)
          fma       $43,$30,$23,$43
          lqd       $27,288($10)
          fma       $44,$30,$24,$44
          lqd       $28,304($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,512($10)
          fma       $42,$34,$26,$42
          lqd       $22,528($10)
          fma       $43,$34,$27,$43
          lqd       $23,544($10)
          fma       $44,$34,$28,$44
          lqd       $24,560($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,768($10)
          fma       $38,$29,$22,$38
          lqd       $26,784($10)
          fma       $39,$29,$23,$39
          lqd       $27,800($10)
          fma       $40,$29,$24,$40
          lqd       $28,816($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,144($9)
          fma       $38,$33,$26,$38
          lqd       $17,400($9)
          fma       $39,$33,$27,$39
          lqd       $18,656($9)
          fma       $40,$33,$28,$40
          lqd       $19,912($9)

          fma       $41,$34,$25,$41
          lqd       $21,1024($10)
          fma       $42,$34,$26,$42
          lqd       $22,1040($10)
          fma       $43,$34,$27,$43
          lqd       $23,1056($10)
          fma       $44,$34,$28,$44
          lqd       $24,1072($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#10
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,1280($10)
```

```
fma      $42,$30,$22,$42
lqd      $26,1296($10)
fma      $43,$30,$23,$43
lqd      $27,1312($10)
fma      $44,$30,$24,$44
lqd      $28,1328($10)

fma      $45,$31,$21,$45
shufb    $33,$16,$16,$13
fma      $46,$31,$22,$46
shufb    $34,$17,$17,$13
fma      $47,$31,$23,$47
shufb    $35,$18,$18,$13
fma      $48,$31,$24,$48
shufb    $36,$19,$19,$13

fma      $49,$32,$21,$49
fma      $50,$32,$22,$50
fma      $51,$32,$23,$51
fma      $52,$32,$24,$52

fma      $37,$33,$25,$37
fma      $38,$33,$26,$38
fma      $39,$33,$27,$39
fma      $40,$33,$28,$40

fma      $41,$34,$25,$41
lqd      $21,1536($10)
fma      $42,$34,$26,$42
lqd      $22,1552($10)
fma      $43,$34,$27,$43
lqd      $23,1568($10)
fma      $44,$34,$28,$44
lqd      $24,1584($10)

fma      $45,$35,$25,$45
shufb    $29,$16,$16,$14
fma      $46,$35,$26,$46
shufb    $30,$17,$17,$14
fma      $47,$35,$27,$47
shufb    $31,$18,$18,$14
fma      $48,$35,$28,$48
shufb    $32,$19,$19,$14

fma      $49,$36,$25,$49
fma      $50,$36,$26,$50
fma      $51,$36,$27,$51
fma      $52,$36,$28,$52


fma      $37,$29,$21,$37
lqd      $25,1792($10)
fma      $38,$29,$22,$38
lqd      $26,1808($10)
fma      $39,$29,$23,$39
lqd      $27,1824($10)
fma      $40,$29,$24,$40
lqd      $28,1840($10)

fma      $41,$30,$21,$41
shufb    $33,$16,$16,$15
fma      $42,$30,$22,$42
shufb    $34,$17,$17,$15
fma      $43,$30,$23,$43
shufb    $35,$18,$18,$15
fma      $44,$30,$24,$44
shufb    $36,$19,$19,$15

fma      $45,$31,$21,$45
fma      $46,$31,$22,$46
fma      $47,$31,$23,$47
fma      $48,$31,$24,$48

fma      $49,$32,$21,$49
fma      $50,$32,$22,$50
fma      $51,$32,$23,$51
fma      $52,$32,$24,$52


fma      $37,$33,$25,$37
lqd      $16,160($9)
fma      $38,$33,$26,$38
lqd      $17,416($9)
fma      $39,$33,$27,$39
lqd      $18,672($9)
fma      $40,$33,$28,$40
lqd      $19,928($9)

fma      $41,$34,$25,$41
lqd      $21,2048($10)
fma      $42,$34,$26,$42
lqd      $22,2064($10)
fma      $43,$34,$27,$43
lqd      $23,2080($10)
fma      $44,$34,$28,$44
lqd      $24,2096($10)

fma      $45,$35,$25,$45
shufb    $29,$16,$16,$12
fma      $46,$35,$26,$46
shufb    $30,$17,$17,$12
fma      $47,$35,$27,$47
shufb    $31,$18,$18,$12
```

```
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#11
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,2304($10)
          fma       $42,$30,$22,$42
          lqd       $26,2320($10)
          fma       $43,$30,$23,$43
          lqd       $27,2336($10)
          fma       $44,$30,$24,$44
          lqd       $28,2352($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,2560($10)
          fma       $42,$34,$26,$42
          lqd       $22,2576($10)
          fma       $43,$34,$27,$43
          lqd       $23,2592($10)
          fma       $44,$34,$28,$44
          lqd       $24,2608($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,2816($10)
          fma       $38,$29,$22,$38
          lqd       $26,2832($10)
          fma       $39,$29,$23,$39
          lqd       $27,2848($10)
          fma       $40,$29,$24,$40
          lqd       $28,2864($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,176($9)
          fma       $38,$33,$26,$38
          lqd       $17,432($9)
          fma       $39,$33,$27,$39
          lqd       $18,688($9)
          fma       $40,$33,$28,$40
          lqd       $19,944($9)

          fma       $41,$34,$25,$41
          lqd       $21,3072($10)
          fma       $42,$34,$26,$42
          lqd       $22,3088($10)
          fma       $43,$34,$27,$43
          lqd       $23,3104($10)
          fma       $44,$34,$28,$44
          lqd       $24,3120($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#12
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,3328($10)
          fma       $42,$30,$22,$42
          lqd       $26,3344($10)
          fma       $43,$30,$23,$43
          lqd       $27,3360($10)
          fma       $44,$30,$24,$44
          lqd       $28,3376($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,3584($10)
          fma       $42,$34,$26,$42
          lqd       $22,3600($10)
          fma       $43,$34,$27,$43
          lqd       $23,3616($10)
          fma       $44,$34,$28,$44
          lqd       $24,3632($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,3840($10)
          fma       $38,$29,$22,$38
          lqd       $26,3856($10)
          fma       $39,$29,$23,$39
          lqd       $27,3872($10)
          fma       $40,$29,$24,$40
          lqd       $28,3888($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15
```

```
            fma      $45,$31,$21,$45
            fma      $46,$31,$22,$46
            fma      $47,$31,$23,$47
            fma      $48,$31,$24,$48

            fma      $49,$32,$21,$49
            fma      $50,$32,$22,$50
            fma      $51,$32,$23,$51
            fma      $52,$32,$24,$52


            fma      $37,$33,$25,$37
            lqd      $16,192($9)
            fma      $38,$33,$26,$38
            lqd      $17,448($9)
            fma      $39,$33,$27,$39
            lqd      $18,704($9)
            fma      $40,$33,$28,$40
            lqd      $19,960($9)

            fma      $41,$34,$25,$41
            lqd      $21,4096($10)
            fma      $42,$34,$26,$42
            lqd      $22,4112($10)
            fma      $43,$34,$27,$43
            lqd      $23,4128($10)
            fma      $44,$34,$28,$44
            lqd      $24,4144($10)

            fma      $45,$35,$25,$45
            shufb    $29,$16,$16,$12
            fma      $46,$35,$26,$46
            shufb    $30,$17,$17,$12
            fma      $47,$35,$27,$47
            shufb    $31,$18,$18,$12
            fma      $48,$35,$28,$48
            shufb    $32,$19,$19,$12

            fma      $49,$36,$25,$49
            fma      $50,$36,$26,$50
            fma      $51,$36,$27,$51
            fma      $52,$36,$28,$52



#13
            fma      $37,$29,$21,$37
            fma      $38,$29,$22,$38
            fma      $39,$29,$23,$39
            fma      $40,$29,$24,$40

            fma      $41,$30,$21,$41
            lqd      $25,4352($10)
            fma      $42,$30,$22,$42
            lqd      $26,4368($10)
            fma      $43,$30,$23,$43
            lqd      $27,4384($10)
            fma      $44,$30,$24,$44
            lqd      $28,4400($10)

            fma      $45,$31,$21,$45
            shufb    $33,$16,$16,$13
            fma      $46,$31,$22,$46
            shufb    $34,$17,$17,$13
            fma      $47,$31,$23,$47
            shufb    $35,$18,$18,$13
            fma      $48,$31,$24,$48
            shufb    $36,$19,$19,$13

            fma      $49,$32,$21,$49
            fma      $50,$32,$22,$50
            fma      $51,$32,$23,$51
            fma      $52,$32,$24,$52

            fma      $37,$33,$25,$37
            fma      $38,$33,$26,$38
            fma      $39,$33,$27,$39
            fma      $40,$33,$28,$40

            fma      $41,$34,$25,$41
            lqd      $21,4608($10)
            fma      $42,$34,$26,$42
            lqd      $22,4624($10)
            fma      $43,$34,$27,$43
            lqd      $23,4640($10)
            fma      $44,$34,$28,$44
            lqd      $24,4656($10)

            fma      $45,$35,$25,$45
            shufb    $29,$16,$16,$14
            fma      $46,$35,$26,$46
            shufb    $30,$17,$17,$14
            fma      $47,$35,$27,$47
            shufb    $31,$18,$18,$14
            fma      $48,$35,$28,$48
            shufb    $32,$19,$19,$14

            fma      $49,$36,$25,$49
            fma      $50,$36,$26,$50
            fma      $51,$36,$27,$51
            fma      $52,$36,$28,$52
```

```
          fma       $37,$29,$21,$37
          lqd       $25,4864($10)
          fma       $38,$29,$22,$38
          lqd       $26,4880($10)
          fma       $39,$29,$23,$39
          lqd       $27,4896($10)
          fma       $40,$29,$24,$40
          lqd       $28,4912($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,208($9)
          fma       $38,$33,$26,$38
          lqd       $17,464($9)
          fma       $39,$33,$27,$39
          lqd       $18,720($9)
          fma       $40,$33,$28,$40
          lqd       $19,976($9)

          fma       $41,$34,$25,$41
          lqd       $21,5120($10)
          fma       $42,$34,$26,$42
          lqd       $22,5136($10)
          fma       $43,$34,$27,$43
          lqd       $23,5152($10)
          fma       $44,$34,$28,$44
          lqd       $24,5168($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#14
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,5376($10)
          fma       $42,$30,$22,$42
          lqd       $26,5392($10)
          fma       $43,$30,$23,$43
          lqd       $27,5408($10)
          fma       $44,$30,$24,$44
          lqd       $28,5424($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,5632($10)
          fma       $42,$34,$26,$42
          lqd       $22,5648($10)
          fma       $43,$34,$27,$43
          lqd      $23,5664($10)
          fma      $44,$34,$28,$44
          lqd      $24,5680($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,5888($10)
          fma      $38,$29,$22,$38
          lqd      $26,5904($10)
          fma      $39,$29,$23,$39
          lqd      $27,5920($10)
          fma      $40,$29,$24,$40
          lqd      $28,5936($10)

          fma      $41,$30,$21,$41
          shufb    $33,$16,$16,$15
          fma      $42,$30,$22,$42
          shufb    $34,$17,$17,$15
          fma      $43,$30,$23,$43
          shufb    $35,$18,$18,$15
          fma      $44,$30,$24,$44
          shufb    $36,$19,$19,$15

          fma      $45,$31,$21,$45
          fma      $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,224($9)
          fma      $38,$33,$26,$38
          lqd      $17,480($9)
          fma      $39,$33,$27,$39
          lqd      $18,736($9)
          fma      $40,$33,$28,$40
          lqd      $19,992($9)

          fma      $41,$34,$25,$41
          lqd      $21,6144($10)
          fma      $42,$34,$26,$42
          lqd      $22,6160($10)
          fma      $43,$34,$27,$43
          lqd      $23,6176($10)
          fma      $44,$34,$28,$44
          lqd      $24,6192($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$12
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$12
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$12
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$12

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#15
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,6400($10)
          fma      $42,$30,$22,$42
          lqd      $26,6416($10)
          fma      $43,$30,$23,$43
          lqd      $27,6432($10)
          fma      $44,$30,$24,$44
          lqd      $28,6448($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,6656($10)
          fma       $42,$34,$26,$42
          lqd       $22,6672($10)
          fma       $43,$34,$27,$43
          lqd       $23,6688($10)
          fma       $44,$34,$28,$44
          lqd       $24,6704($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,6912($10)
          fma       $38,$29,$22,$38
          lqd       $26,6928($10)
          fma       $39,$29,$23,$39
          lqd       $27,6944($10)
          fma       $40,$29,$24,$40
          lqd       $28,6960($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,240($9)
          fma       $38,$33,$26,$38
          lqd       $17,496($9)
          fma       $39,$33,$27,$39
          lqd       $18,752($9)
          fma       $40,$33,$28,$40
          lqd       $19,1008($9)

          fma       $41,$34,$25,$41
          lqd       $21,7168($10)
          fma       $42,$34,$26,$42
          lqd       $22,7184($10)
          fma       $43,$34,$27,$43
          lqd       $23,7200($10)
          fma       $44,$34,$28,$44
          lqd       $24,7216($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


#16
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,7424($10)
          fma       $42,$30,$22,$42
          lqd       $26,7440($10)
          fma       $43,$30,$23,$43
          lqd       $27,7456($10)
          fma       $44,$30,$24,$44
          lqd       $28,7472($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,7680($10)
          fma       $42,$34,$26,$42
          lqd       $22,7696($10)
          fma       $43,$34,$27,$43
          lqd       $23,7712($10)
          fma       $44,$34,$28,$44
          lqd       $24,7728($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,7936($10)
          fma       $38,$29,$22,$38
          lqd       $26,7952($10)
          fma       $39,$29,$23,$39
          lqd       $27,7968($10)
          fma       $40,$29,$24,$40
          lqd       $28,7984($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          rotqbyi   $9,$59,0
          fma       $46,$31,$22,$46
#         ai        $10,$4,64
          shufb     $10,$55,$55,$13
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,0($9)
          fma       $38,$33,$26,$38
          lqd       $17,256($9)
          fma       $39,$33,$27,$39
          lqd       $18,512($9)
          fma       $40,$33,$28,$40
          lqd       $19,768($9)

          fma       $41,$34,$25,$41
          lqd       $21,0($10)
          fma       $42,$34,$26,$42
          lqd       $22,16($10)
          fma       $43,$34,$27,$43
          lqd       $23,32($10)
          fma       $44,$34,$28,$44
          lqd       $24,48($10)
          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          stqd      $37,0($11)
          fma       $50,$36,$26,$50
          stqd      $38,16($11)
          fma       $51,$36,$27,$51
          stqd      $39,32($11)
          fma       $52,$36,$28,$52
          stqd      $40,48($11)




#N16-N31




#1
          fma       $61,$29,$21,$61
          lqd       $67,288($77)
          fma       $62,$29,$22,$62
          lqd       $68,304($77)
          fma       $63,$29,$23,$63
          lqd       $69,512($77)
          fma       $64,$29,$24,$64
          lqd       $70,528($77)

          fma       $65,$30,$21,$65
          lqd       $71,544($77)
          fma       $66,$30,$22,$66
          lqd       $72,560($77)
          fma       $67,$30,$23,$67
          lqd       $73,768($77)
          fma       $68,$30,$24,$68
          lqd       $74,784($77)

          fma       $69,$31,$21,$69
          lqd       $75,800($77)
          fma       $70,$31,$22,$70
          lqd       $76,816($77)
          fma       $71,$31,$23,$71
          lqd       $25,256($10)
          fma       $72,$31,$24,$72
          lqd       $26,272($10)

          fma       $73,$32,$21,$73
          lqd       $27,288($10)
          fma       $74,$32,$22,$74
          shufb     $33,$16,$16,$13
          fma       $75,$32,$23,$75
          lqd       $28,304($10)
          fma       $76,$32,$24,$76
          shufb     $34,$17,$17,$13

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          shufb     $35,$18,$18,$13
          fma       $64,$33,$28,$64
          shufb     $36,$19,$19,$13

          fma       $65,$34,$25,$65
          lqd       $21,512($10)
          fma       $66,$34,$26,$66
          lqd       $22,528($10)
          fma       $67,$34,$27,$67
          lqd       $23,544($10)
          fma       $68,$34,$28,$68
          lqd       $24,560($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,768($10)
          fma       $62,$29,$22,$62
          lqd       $26,784($10)
          fma       $63,$29,$23,$63
          lqd       $27,800($10)
          fma       $64,$29,$24,$64
```

```
        lqd     $28,816($10)

        fma     $65,$30,$21,$65
        shufb   $33,$16,$16,$15
        fma     $66,$30,$22,$66
        shufb   $34,$17,$17,$15
        fma     $67,$30,$23,$67
        shufb   $35,$18,$18,$15
        fma     $68,$30,$24,$68
        shufb   $36,$19,$19,$15

        fma     $69,$31,$21,$69
        fma     $70,$31,$22,$70
        fma     $71,$31,$23,$71
        fma     $72,$31,$24,$72

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76


        fma     $61,$33,$25,$61
        lqd     $16,16($9)
        fma     $62,$33,$26,$62
        lqd     $17,272($9)
        fma     $63,$33,$27,$63
        lqd     $18,528($9)
        fma     $64,$33,$28,$64
        lqd     $19,784($9)

        fma     $65,$34,$25,$65
        lqd     $21,1024($10)
        fma     $66,$34,$26,$66
        lqd     $22,1040($10)
        fma     $67,$34,$27,$67
        lqd     $23,1056($10)
        fma     $68,$34,$28,$68
        lqd     $24,1072($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76



#2
        fma     $61,$29,$21,$61
        stqd    $41,256($11)
        fma     $62,$29,$22,$62
        stqd    $42,272($11)
        fma     $63,$29,$23,$63
        stqd    $43,288($11)
        fma     $64,$29,$24,$64
        stqd    $44,304($11)

        fma     $65,$30,$21,$65
        lqd     $25,1280($10)
        fma     $66,$30,$22,$66
        lqd     $26,1296($10)
        fma     $67,$30,$23,$67
        lqd     $27,1312($10)
        fma     $68,$30,$24,$68
        lqd     $28,1328($10)

        fma     $69,$31,$21,$69
        shufb   $33,$16,$16,$13
        fma     $70,$31,$22,$70
        shufb   $34,$17,$17,$13
        fma     $71,$31,$23,$71
        shufb   $35,$18,$18,$13
        fma     $72,$31,$24,$72
        shufb   $36,$19,$19,$13

        fma     $73,$32,$21,$73
        stqd    $45,512($11)
        fma     $74,$32,$22,$74
        stqd    $46,528($11)
        fma     $75,$32,$23,$75
        stqd    $47,544($11)
        fma     $76,$32,$24,$76
        stqd    $48,560($11)

        fma     $61,$33,$25,$61
        stqd    $49,768($11)
        fma     $62,$33,$26,$62
        stqd    $50,784($11)
        fma     $63,$33,$27,$63
        stqd    $51,800($11)
        fma     $64,$33,$28,$64
        stqd    $52,816($11)
        ai      $11,$11,128
        lnop
```

```
            fma     $65,$34,$25,$65
            lqd     $21,1536($10)
            fma     $66,$34,$26,$66
            lqd     $22,1552($10)
            fma     $67,$34,$27,$67
            lqd     $23,1568($10)
            fma     $68,$34,$28,$68
            lqd     $24,1584($10)

            fma     $69,$35,$25,$69
            shufb   $29,$16,$16,$14
            fma     $70,$35,$26,$70
            shufb   $30,$17,$17,$14
            fma     $71,$35,$27,$71
            shufb   $31,$18,$18,$14
            fma     $72,$35,$28,$72
            shufb   $32,$19,$19,$14

            fma     $73,$36,$25,$73
            fma     $74,$36,$26,$74
            fma     $75,$36,$27,$75
            fma     $76,$36,$28,$76


            fma     $61,$29,$21,$61
            lqd     $25,1792($10)
            fma     $62,$29,$22,$62
            lqd     $26,1808($10)
            fma     $63,$29,$23,$63
            lqd     $27,1824($10)
            fma     $64,$29,$24,$64
            lqd     $28,1840($10)

            fma     $65,$30,$21,$65
            shufb   $33,$16,$16,$15
            fma     $66,$30,$22,$66
            shufb   $34,$17,$17,$15
            fma     $67,$30,$23,$67
            shufb   $35,$18,$18,$15
            fma     $68,$30,$24,$68
            shufb   $36,$19,$19,$15

            fma     $69,$31,$21,$69
            fma     $70,$31,$22,$70
            fma     $71,$31,$23,$71
            fma     $72,$31,$24,$72

            fma     $73,$32,$21,$73
            fma     $74,$32,$22,$74
            fma     $75,$32,$23,$75
            fma     $76,$32,$24,$76


            fma     $61,$33,$25,$61
            lqd     $16,32($9)
            fma     $62,$33,$26,$62
            lqd     $17,288($9)
            fma     $63,$33,$27,$63
            lqd     $18,544($9)
            fma     $64,$33,$28,$64
            lqd     $19,800($9)

            fma     $65,$34,$25,$65
            lqd     $21,2048($10)
            fma     $66,$34,$26,$66
            lqd     $22,2064($10)
            fma     $67,$34,$27,$67
            lqd     $23,2080($10)
            fma     $68,$34,$28,$68
            lqd     $24,2096($10)

            fma     $69,$35,$25,$69
            shufb   $29,$16,$16,$12
            fma     $70,$35,$26,$70
            shufb   $30,$17,$17,$12
            fma     $71,$35,$27,$71
            shufb   $31,$18,$18,$12
            fma     $72,$35,$28,$72
            shufb   $32,$19,$19,$12

            fma     $73,$36,$25,$73
            fma     $74,$36,$26,$74
            fma     $75,$36,$27,$75
            fma     $76,$36,$28,$76



#3
            fma     $61,$29,$21,$61
            fma     $62,$29,$22,$62
            fma     $63,$29,$23,$63
            fma     $64,$29,$24,$64

            fma     $65,$30,$21,$65
            lqd     $25,2304($10)
            fma     $66,$30,$22,$66
            lqd     $26,2320($10)
            fma     $67,$30,$23,$67
            lqd     $27,2336($10)
            fma     $68,$30,$24,$68
            lqd     $28,2352($10)
```

```
fma      $69,$31,$21,$69
shufb    $33,$16,$16,$13
fma      $70,$31,$22,$70
shufb    $34,$17,$17,$13
fma      $71,$31,$23,$71
shufb    $35,$18,$18,$13
fma      $72,$31,$24,$72
shufb    $36,$19,$19,$13

fma      $73,$32,$21,$73
lqd      $37,0($11)
fma      $74,$32,$22,$74
lqd      $38,16($11)
fma      $75,$32,$23,$75
lqd      $39,32($11)
fma      $76,$32,$24,$76
lqd      $40,48($11)

fma      $61,$33,$25,$61
lqd      $41,256($11)
fma      $62,$33,$26,$62
lqd      $42,272($11)
fma      $63,$33,$27,$63
fma      $64,$33,$28,$64

fma      $65,$34,$25,$65
lqd      $21,2560($10)
fma      $66,$34,$26,$66
lqd      $22,2576($10)
fma      $67,$34,$27,$67
lqd      $23,2592($10)
fma      $68,$34,$28,$68
lqd      $24,2608($10)

fma      $69,$35,$25,$69
shufb    $29,$16,$16,$14
fma      $70,$35,$26,$70
shufb    $30,$17,$17,$14
fma      $71,$35,$27,$71
shufb    $31,$18,$18,$14
fma      $72,$35,$28,$72
shufb    $32,$19,$19,$14

fma      $73,$36,$25,$73
fma      $74,$36,$26,$74
fma      $75,$36,$27,$75
fma      $76,$36,$28,$76


fma      $61,$29,$21,$61
lqd      $25,2816($10)
fma      $62,$29,$22,$62
lqd      $26,2832($10)
fma      $63,$29,$23,$63
lqd      $27,2848($10)
fma      $64,$29,$24,$64
lqd      $28,2864($10)

fma      $65,$30,$21,$65
shufb    $33,$16,$16,$15
fma      $66,$30,$22,$66
shufb    $34,$17,$17,$15
fma      $67,$30,$23,$67
shufb    $35,$18,$18,$15
fma      $68,$30,$24,$68
shufb    $36,$19,$19,$15

fma      $69,$31,$21,$69
fma      $70,$31,$22,$70
fma      $71,$31,$23,$71
fma      $72,$31,$24,$72

fma      $73,$32,$21,$73
fma      $74,$32,$22,$74
fma      $75,$32,$23,$75
fma      $76,$32,$24,$76


fma      $61,$33,$25,$61
lqd      $16,48($9)
fma      $62,$33,$26,$62
lqd      $17,304($9)
fma      $63,$33,$27,$63
lqd      $18,560($9)
fma      $64,$33,$28,$64
lqd      $19,816($9)

fma      $65,$34,$25,$65
lqd      $21,3072($10)
fma      $66,$34,$26,$66
lqd      $22,3088($10)
fma      $67,$34,$27,$67
lqd      $23,3104($10)
fma      $68,$34,$28,$68
lqd      $24,3120($10)

fma      $69,$35,$25,$69
shufb    $29,$16,$16,$12
fma      $70,$35,$26,$70
shufb    $30,$17,$17,$12
fma      $71,$35,$27,$71
shufb    $31,$18,$18,$12
fma      $72,$35,$28,$72
```

```
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#4
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,3328($10)
          fma       $66,$30,$22,$66
          lqd       $26,3344($10)
          fma       $67,$30,$23,$67
          lqd       $27,3360($10)
          fma       $68,$30,$24,$68
          lqd       $28,3376($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,3584($10)
          fma       $66,$34,$26,$66
          lqd       $22,3600($10)
          fma       $67,$34,$27,$67
          lqd       $23,3616($10)
          fma       $68,$34,$28,$68
          lqd       $24,3632($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,3840($10)
          fma       $62,$29,$22,$62
          lqd       $26,3856($10)
          fma       $63,$29,$23,$63
          lqd       $27,3872($10)
          fma       $64,$29,$24,$64
          lqd       $28,3888($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,64($9)
          fma       $62,$33,$26,$62
          lqd       $17,320($9)
          fma       $63,$33,$27,$63
          lqd       $18,576($9)
          fma       $64,$33,$28,$64
          lqd       $19,832($9)

          fma       $65,$34,$25,$65
          lqd       $21,4096($10)
          fma       $66,$34,$26,$66
          lqd       $22,4112($10)
          fma       $67,$34,$27,$67
          lqd       $23,4128($10)
          fma       $68,$34,$28,$68
          lqd       $24,4144($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#5
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,4352($10)
          fma       $66,$30,$22,$66
          lqd       $26,4368($10)
          fma       $67,$30,$23,$67
          lqd       $27,4384($10)
          fma       $68,$30,$24,$68
          lqd       $28,4400($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,4608($10)
          fma       $66,$34,$26,$66
          lqd       $22,4624($10)
          fma       $67,$34,$27,$67
          lqd       $23,4640($10)
          fma       $68,$34,$28,$68
          lqd       $24,4656($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,4864($10)
          fma       $62,$29,$22,$62
          lqd       $26,4880($10)
          fma       $63,$29,$23,$63
          lqd       $27,4896($10)
          fma       $64,$29,$24,$64
          lqd       $28,4912($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15
```

```
        fma     $69,$31,$21,$69
        fma     $70,$31,$22,$70
        fma     $71,$31,$23,$71
        fma     $72,$31,$24,$72

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76


        fma     $61,$33,$25,$61
        lqd     $16,80($9)
        fma     $62,$33,$26,$62
        lqd     $17,336($9)
        fma     $63,$33,$27,$63
        lqd     $18,592($9)
        fma     $64,$33,$28,$64
        lqd     $19,848($9)

        fma     $65,$34,$25,$65
        lqd     $21,5120($10)
        fma     $66,$34,$26,$66
        lqd     $22,5136($10)
        fma     $67,$34,$27,$67
        lqd     $23,5152($10)
        fma     $68,$34,$28,$68
        lqd     $24,5168($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76



#6
        fma     $61,$29,$21,$61
        fma     $62,$29,$22,$62
        fma     $63,$29,$23,$63
        fma     $64,$29,$24,$64

        fma     $65,$30,$21,$65
        lqd     $25,5376($10)
        fma     $66,$30,$22,$66
        lqd     $26,5392($10)
        fma     $67,$30,$23,$67
        lqd     $27,5408($10)
        fma     $68,$30,$24,$68
        lqd     $28,5424($10)

        fma     $69,$31,$21,$69
        shufb   $33,$16,$16,$13
        fma     $70,$31,$22,$70
        shufb   $34,$17,$17,$13
        fma     $71,$31,$23,$71
        shufb   $35,$18,$18,$13
        fma     $72,$31,$24,$72
        shufb   $36,$19,$19,$13

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76

        fma     $61,$33,$25,$61
        fma     $62,$33,$26,$62
        fma     $63,$33,$27,$63
        fma     $64,$33,$28,$64

        fma     $65,$34,$25,$65
        lqd     $21,5632($10)
        fma     $66,$34,$26,$66
        lqd     $22,5648($10)
        fma     $67,$34,$27,$67
        lqd     $23,5664($10)
        fma     $68,$34,$28,$68
        lqd     $24,5680($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$14
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$14
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$14
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$14

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76
```

```
          fma      $61,$29,$21,$61
          lqd      $25,5888($10)
          fma      $62,$29,$22,$62
          lqd      $26,5904($10)
          fma      $63,$29,$23,$63
          lqd      $27,5920($10)
          fma      $64,$29,$24,$64
          lqd      $28,5936($10)

          fma      $65,$30,$21,$65
          shufb    $33,$16,$16,$15
          fma      $66,$30,$22,$66
          shufb    $34,$17,$17,$15
          fma      $67,$30,$23,$67
          shufb    $35,$18,$18,$15
          fma      $68,$30,$24,$68
          shufb    $36,$19,$19,$15

          fma      $69,$31,$21,$69
          fma      $70,$31,$22,$70
          fma      $71,$31,$23,$71
          fma      $72,$31,$24,$72

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76


          fma      $61,$33,$25,$61
          lqd      $16,96($9)
          fma      $62,$33,$26,$62
          lqd      $17,352($9)
          fma      $63,$33,$27,$63
          lqd      $18,608($9)
          fma      $64,$33,$28,$64
          lqd      $19,864($9)

          fma      $65,$34,$25,$65
          lqd      $21,6144($10)
          fma      $66,$34,$26,$66
          lqd      $22,6160($10)
          fma      $67,$34,$27,$67
          lqd      $23,6176($10)
          fma      $68,$34,$28,$68
          lqd      $24,6192($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$12
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$12
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$12
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$12

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76



#7
          fma      $61,$29,$21,$61
          fma      $62,$29,$22,$62
          fma      $63,$29,$23,$63
          fma      $64,$29,$24,$64

          fma      $65,$30,$21,$65
          lqd      $25,6400($10)
          fma      $66,$30,$22,$66
          lqd      $26,6416($10)
          fma      $67,$30,$23,$67
          lqd      $27,6432($10)
          fma      $68,$30,$24,$68
          lqd      $28,6448($10)

          fma      $69,$31,$21,$69
          shufb    $33,$16,$16,$13
          fma      $70,$31,$22,$70
          shufb    $34,$17,$17,$13
          fma      $71,$31,$23,$71
          shufb    $35,$18,$18,$13
          fma      $72,$31,$24,$72
          shufb    $36,$19,$19,$13

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76

          fma      $61,$33,$25,$61
          fma      $62,$33,$26,$62
          fma      $63,$33,$27,$63
          fma      $64,$33,$28,$64

          fma      $65,$34,$25,$65
          lqd      $21,6656($10)
          fma      $66,$34,$26,$66
          lqd      $22,6672($10)
          fma      $67,$34,$27,$67
          lqd      $23,6688($10)
          fma       $68,$34,$28,$68
          lqd       $24,6704($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,6912($10)
          fma       $62,$29,$22,$62
          lqd       $26,6928($10)
          fma       $63,$29,$23,$63
          lqd       $27,6944($10)
          fma       $64,$29,$24,$64
          lqd       $28,6960($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,112($9)
          fma       $62,$33,$26,$62
          lqd       $17,368($9)
          fma       $63,$33,$27,$63
          lqd       $18,624($9)
          fma       $64,$33,$28,$64
          lqd       $19,880($9)

          fma       $65,$34,$25,$65
          lqd       $21,7168($10)
          fma       $66,$34,$26,$66
          lqd       $22,7184($10)
          fma       $67,$34,$27,$67
          lqd       $23,7200($10)
          fma       $68,$34,$28,$68
          lqd       $24,7216($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#8
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,7424($10)
          fma       $66,$30,$22,$66
          lqd       $26,7440($10)
          fma       $67,$30,$23,$67
          lqd       $27,7456($10)
          fma       $68,$30,$24,$68
          lqd       $28,7472($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
```

```
            shufb    $36,$19,$19,$13

            fma      $73,$32,$21,$73
            fma      $74,$32,$22,$74
            fma      $75,$32,$23,$75
            fma      $76,$32,$24,$76

            fma      $61,$33,$25,$61
            fma      $62,$33,$26,$62
            fma      $63,$33,$27,$63
            fma      $64,$33,$28,$64

            fma      $65,$34,$25,$65
            lqd      $21,7680($10)
            fma      $66,$34,$26,$66
            lqd      $22,7696($10)
            fma      $67,$34,$27,$67
            lqd      $23,7712($10)
            fma      $68,$34,$28,$68
            lqd      $24,7728($10)

            fma      $69,$35,$25,$69
            shufb    $29,$16,$16,$14
            fma      $70,$35,$26,$70
            shufb    $30,$17,$17,$14
            fma      $71,$35,$27,$71
            shufb    $31,$18,$18,$14
            fma      $72,$35,$28,$72
            shufb    $32,$19,$19,$14

            fma      $73,$36,$25,$73
            fma      $74,$36,$26,$74
            fma      $75,$36,$27,$75
            fma      $76,$36,$28,$76


            fma      $61,$29,$21,$61
            lqd      $25,7936($10)
            fma      $62,$29,$22,$62
            lqd      $26,7952($10)
            fma      $63,$29,$23,$63
            lqd      $27,7968($10)
            fma      $64,$29,$24,$64
            lqd      $28,7984($10)

            fma      $65,$30,$21,$65
            shufb    $33,$16,$16,$15
            fma      $66,$30,$22,$66
            shufb    $34,$17,$17,$15
            fma      $67,$30,$23,$67
            shufb    $35,$18,$18,$15
            fma      $68,$30,$24,$68
            shufb    $36,$19,$19,$15

            fma      $69,$31,$21,$69
            fma      $70,$31,$22,$70
            fma      $71,$31,$23,$71
#           a        $10,$10,$78
            shufb    $10,$58,$58,$13
            fma      $72,$31,$24,$72
            lnop

            fma      $73,$32,$21,$73
            fma      $74,$32,$22,$74
            fma      $75,$32,$23,$75
            fma      $76,$32,$24,$76


            fma      $61,$33,$25,$61
            lqd      $16,128($9)
            fma      $62,$33,$26,$62
            lqd      $17,384($9)
            fma      $63,$33,$27,$63
            lqd      $18,640($9)
            fma      $64,$33,$28,$64
            lqd      $19,896($9)

            fma      $65,$34,$25,$65
            lqd      $21,0($10)
            fma      $66,$34,$26,$66
            lqd      $22,16($10)
            fma      $67,$34,$27,$67
            lqd      $23,32($10)
            fma      $68,$34,$28,$68
            lqd      $24,48($10)

            fma      $69,$35,$25,$69
            shufb    $29,$16,$16,$12
            fma      $70,$35,$26,$70
            shufb    $30,$17,$17,$12
            fma      $71,$35,$27,$71
            shufb    $31,$18,$18,$12
            fma      $72,$35,$28,$72
            shufb    $32,$19,$19,$12

            fma      $73,$36,$25,$73
            fma      $74,$36,$26,$74
            fma      $75,$36,$27,$75
            fma      $76,$36,$28,$76


#9
```

```
fma      $61,$29,$21,$61
fma      $62,$29,$22,$62
fma      $63,$29,$23,$63
fma      $64,$29,$24,$64

fma      $65,$30,$21,$65
lqd      $25,256($10)
fma      $66,$30,$22,$66
lqd      $26,272($10)
fma      $67,$30,$23,$67
lqd      $27,288($10)
fma      $68,$30,$24,$68
lqd      $28,304($10)

fma      $69,$31,$21,$69
shufb    $33,$16,$16,$13
fma      $70,$31,$22,$70
shufb    $34,$17,$17,$13
fma      $71,$31,$23,$71
shufb    $35,$18,$18,$13
fma      $72,$31,$24,$72
shufb    $36,$19,$19,$13

fma      $73,$32,$21,$73
fma      $74,$32,$22,$74
fma      $75,$32,$23,$75
fma      $76,$32,$24,$76

fma      $61,$33,$25,$61
fma      $62,$33,$26,$62
fma      $63,$33,$27,$63
fma      $64,$33,$28,$64

fma      $65,$34,$25,$65
lqd      $21,512($10)
fma      $66,$34,$26,$66
lqd      $22,528($10)
fma      $67,$34,$27,$67
lqd      $23,544($10)
fma      $68,$34,$28,$68
lqd      $24,560($10)

fma      $69,$35,$25,$69
shufb    $29,$16,$16,$14
fma      $70,$35,$26,$70
shufb    $30,$17,$17,$14
fma      $71,$35,$27,$71
shufb    $31,$18,$18,$14
fma      $72,$35,$28,$72
shufb    $32,$19,$19,$14

fma      $73,$36,$25,$73
fma      $74,$36,$26,$74
fma      $75,$36,$27,$75
fma      $76,$36,$28,$76


fma      $61,$29,$21,$61
lqd      $25,768($10)
fma      $62,$29,$22,$62
lqd      $26,784($10)
fma      $63,$29,$23,$63
lqd      $27,800($10)
fma      $64,$29,$24,$64
lqd      $28,816($10)

fma      $65,$30,$21,$65
shufb    $33,$16,$16,$15
fma      $66,$30,$22,$66
shufb    $34,$17,$17,$15
fma      $67,$30,$23,$67
shufb    $35,$18,$18,$15
fma      $68,$30,$24,$68
shufb    $36,$19,$19,$15

fma      $69,$31,$21,$69
fma      $70,$31,$22,$70
fma      $71,$31,$23,$71
fma      $72,$31,$24,$72

fma      $73,$32,$21,$73
fma      $74,$32,$22,$74
fma      $75,$32,$23,$75
fma      $76,$32,$24,$76


fma      $61,$33,$25,$61
lqd      $16,144($9)
fma      $62,$33,$26,$62
lqd      $17,400($9)
fma      $63,$33,$27,$63
lqd      $18,656($9)
fma      $64,$33,$28,$64
lqd      $19,912($9)

fma      $65,$34,$25,$65
lqd      $21,1024($10)
fma      $66,$34,$26,$66
lqd      $22,1040($10)
fma      $67,$34,$27,$67
lqd      $23,1056($10)
fma      $68,$34,$28,$68
lqd      $24,1072($10)
```

```
          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#10
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,1280($10)
          fma       $66,$30,$22,$66
          lqd       $26,1296($10)
          fma       $67,$30,$23,$67
          lqd       $27,1312($10)
          fma       $68,$30,$24,$68
          lqd       $28,1328($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,1536($10)
          fma       $66,$34,$26,$66
          lqd       $22,1552($10)
          fma       $67,$34,$27,$67
          lqd       $23,1568($10)
          fma       $68,$34,$28,$68
          lqd       $24,1584($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,1792($10)
          fma       $62,$29,$22,$62
          lqd       $26,1808($10)
          fma       $63,$29,$23,$63
          lqd       $27,1824($10)
          fma       $64,$29,$24,$64
          lqd       $28,1840($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76
```

```
        fma     $61,$33,$25,$61
        lqd     $16,160($9)
        fma     $62,$33,$26,$62
        lqd     $17,416($9)
        fma     $63,$33,$27,$63
        lqd     $18,672($9)
        fma     $64,$33,$28,$64
        lqd     $19,928($9)

        fma     $65,$34,$25,$65
        lqd     $21,2048($10)
        fma     $66,$34,$26,$66
        lqd     $22,2064($10)
        fma     $67,$34,$27,$67
        lqd     $23,2080($10)
        fma     $68,$34,$28,$68
        lqd     $24,2096($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76



#11
        fma     $61,$29,$21,$61
        fma     $62,$29,$22,$62
        fma     $63,$29,$23,$63
        fma     $64,$29,$24,$64

        fma     $65,$30,$21,$65
        lqd     $25,2304($10)
        fma     $66,$30,$22,$66
        lqd     $26,2320($10)
        fma     $67,$30,$23,$67
        lqd     $27,2336($10)
        fma     $68,$30,$24,$68
        lqd     $28,2352($10)

        fma     $69,$31,$21,$69
        shufb   $33,$16,$16,$13
        fma     $70,$31,$22,$70
        shufb   $34,$17,$17,$13
        fma     $71,$31,$23,$71
        shufb   $35,$18,$18,$13
        fma     $72,$31,$24,$72
        shufb   $36,$19,$19,$13

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76

        fma     $61,$33,$25,$61
        fma     $62,$33,$26,$62
        fma     $63,$33,$27,$63
        fma     $64,$33,$28,$64

        fma     $65,$34,$25,$65
        lqd     $21,2560($10)
        fma     $66,$34,$26,$66
        lqd     $22,2576($10)
        fma     $67,$34,$27,$67
        lqd     $23,2592($10)
        fma     $68,$34,$28,$68
        lqd     $24,2608($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$14
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$14
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$14
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$14

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76


        fma     $61,$29,$21,$61
        lqd     $25,2816($10)
        fma     $62,$29,$22,$62
        lqd     $26,2832($10)
        fma     $63,$29,$23,$63
        lqd     $27,2848($10)
        fma     $64,$29,$24,$64
        lqd     $28,2864($10)

        fma     $65,$30,$21,$65
```

```
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,176($9)
          fma       $62,$33,$26,$62
          lqd       $17,432($9)
          fma       $63,$33,$27,$63
          lqd       $18,688($9)
          fma       $64,$33,$28,$64
          lqd       $19,944($9)

          fma       $65,$34,$25,$65
          lqd       $21,3072($10)
          fma       $66,$34,$26,$66
          lqd       $22,3088($10)
          fma       $67,$34,$27,$67
          lqd       $23,3104($10)
          fma       $68,$34,$28,$68
          lqd       $24,3120($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#12
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,3328($10)
          fma       $66,$30,$22,$66
          lqd       $26,3344($10)
          fma       $67,$30,$23,$67
          lqd       $27,3360($10)
          fma       $68,$30,$24,$68
          lqd       $28,3376($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,3584($10)
          fma       $66,$34,$26,$66
          lqd       $22,3600($10)
          fma       $67,$34,$27,$67
          lqd       $23,3616($10)
          fma       $68,$34,$28,$68
          lqd       $24,3632($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,3840($10)
          fma       $62,$29,$22,$62
          lqd       $26,3856($10)
          fma       $63,$29,$23,$63
          lqd       $27,3872($10)
          fma       $64,$29,$24,$64
          lqd       $28,3888($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,192($9)
          fma       $62,$33,$26,$62
          lqd       $17,448($9)
          fma       $63,$33,$27,$63
          lqd       $18,704($9)
          fma       $64,$33,$28,$64
          lqd       $19,960($9)

          fma       $65,$34,$25,$65
          lqd       $21,4096($10)
          fma       $66,$34,$26,$66
          lqd       $22,4112($10)
          fma       $67,$34,$27,$67
          lqd       $23,4128($10)
          fma       $68,$34,$28,$68
          lqd       $24,4144($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#13
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,4352($10)
          fma       $66,$30,$22,$66
          lqd       $26,4368($10)
          fma       $67,$30,$23,$67
          lqd       $27,4384($10)
          fma       $68,$30,$24,$68
          lqd       $28,4400($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
```

```
            fma       $64,$33,$28,$64

            fma       $65,$34,$25,$65
            lqd       $21,4608($10)
            fma       $66,$34,$26,$66
            lqd       $22,4624($10)
            fma       $67,$34,$27,$67
            lqd       $23,4640($10)
            fma       $68,$34,$28,$68
            lqd       $24,4656($10)

            fma       $69,$35,$25,$69
            shufb     $29,$16,$16,$14
            fma       $70,$35,$26,$70
            shufb     $30,$17,$17,$14
            fma       $71,$35,$27,$71
            shufb     $31,$18,$18,$14
            fma       $72,$35,$28,$72
            shufb     $32,$19,$19,$14

            fma       $73,$36,$25,$73
            fma       $74,$36,$26,$74
            fma       $75,$36,$27,$75
            fma       $76,$36,$28,$76


            fma       $61,$29,$21,$61
            lqd       $25,4864($10)
            fma       $62,$29,$22,$62
            lqd       $26,4880($10)
            fma       $63,$29,$23,$63
            lqd       $27,4896($10)
            fma       $64,$29,$24,$64
            lqd       $28,4912($10)

            fma       $65,$30,$21,$65
            shufb     $33,$16,$16,$15
            fma       $66,$30,$22,$66
            shufb     $34,$17,$17,$15
            fma       $67,$30,$23,$67
            shufb     $35,$18,$18,$15
            fma       $68,$30,$24,$68
            shufb     $36,$19,$19,$15

            fma       $69,$31,$21,$69
            fma       $70,$31,$22,$70
            fma       $71,$31,$23,$71
            fma       $72,$31,$24,$72

            fma       $73,$32,$21,$73
            fma       $74,$32,$22,$74
            fma       $75,$32,$23,$75
            fma       $76,$32,$24,$76


            fma       $61,$33,$25,$61
            lqd       $16,208($9)
            fma       $62,$33,$26,$62
            lqd       $17,464($9)
            fma       $63,$33,$27,$63
            lqd       $18,720($9)
            fma       $64,$33,$28,$64
            lqd       $19,976($9)

            fma       $65,$34,$25,$65
            lqd       $21,5120($10)
            fma       $66,$34,$26,$66
            lqd       $22,5136($10)
            fma       $67,$34,$27,$67
            lqd       $23,5152($10)
            fma       $68,$34,$28,$68
            lqd       $24,5168($10)

            fma       $69,$35,$25,$69
            shufb     $29,$16,$16,$12
            fma       $70,$35,$26,$70
            shufb     $30,$17,$17,$12
            fma       $71,$35,$27,$71
            shufb     $31,$18,$18,$12
            fma       $72,$35,$28,$72
            shufb     $32,$19,$19,$12

            fma       $73,$36,$25,$73
            fma       $74,$36,$26,$74
            fma       $75,$36,$27,$75
            fma       $76,$36,$28,$76



#14
            fma       $61,$29,$21,$61
            fma       $62,$29,$22,$62
            fma       $63,$29,$23,$63
            fma       $64,$29,$24,$64

            fma       $65,$30,$21,$65
            lqd       $25,5376($10)
            fma       $66,$30,$22,$66
            lqd       $26,5392($10)
            fma       $67,$30,$23,$67
            lqd       $27,5408($10)
            fma       $68,$30,$24,$68
            lqd       $28,5424($10)
```

```
fma       $69,$31,$21,$69
shufb     $33,$16,$16,$13
fma       $70,$31,$22,$70
shufb     $34,$17,$17,$13
fma       $71,$31,$23,$71
shufb     $35,$18,$18,$13
fma       $72,$31,$24,$72
shufb     $36,$19,$19,$13

fma       $73,$32,$21,$73
fma       $74,$32,$22,$74
fma       $75,$32,$23,$75
fma       $76,$32,$24,$76

fma       $61,$33,$25,$61
fma       $62,$33,$26,$62
fma       $63,$33,$27,$63
fma       $64,$33,$28,$64

fma       $65,$34,$25,$65
lqd       $21,5632($10)
fma       $66,$34,$26,$66
lqd       $22,5648($10)
fma       $67,$34,$27,$67
lqd       $23,5664($10)
fma       $68,$34,$28,$68
lqd       $24,5680($10)

fma       $69,$35,$25,$69
shufb     $29,$16,$16,$14
fma       $70,$35,$26,$70
shufb     $30,$17,$17,$14
fma       $71,$35,$27,$71
shufb     $31,$18,$18,$14
fma       $72,$35,$28,$72
shufb     $32,$19,$19,$14

fma       $73,$36,$25,$73
fma       $74,$36,$26,$74
fma       $75,$36,$27,$75
fma       $76,$36,$28,$76


fma       $61,$29,$21,$61
lqd       $25,5888($10)
fma       $62,$29,$22,$62
lqd       $26,5904($10)
fma       $63,$29,$23,$63
lqd       $27,5920($10)
fma       $64,$29,$24,$64
lqd       $28,5936($10)

fma       $65,$30,$21,$65
shufb     $33,$16,$16,$15
fma       $66,$30,$22,$66
shufb     $34,$17,$17,$15
fma       $67,$30,$23,$67
shufb     $35,$18,$18,$15
fma       $68,$30,$24,$68
shufb     $36,$19,$19,$15

fma       $69,$31,$21,$69
fma       $70,$31,$22,$70
fma       $71,$31,$23,$71
fma       $72,$31,$24,$72

fma       $73,$32,$21,$73
fma       $74,$32,$22,$74
fma       $75,$32,$23,$75
fma       $76,$32,$24,$76


fma       $61,$33,$25,$61
lqd       $16,224($9)
fma       $62,$33,$26,$62
lqd       $17,480($9)
fma       $63,$33,$27,$63
lqd       $18,736($9)
fma       $64,$33,$28,$64
lqd       $19,992($9)

fma       $65,$34,$25,$65
lqd       $21,6144($10)
fma       $66,$34,$26,$66
lqd       $22,6160($10)
fma       $67,$34,$27,$67
lqd       $23,6176($10)
fma       $68,$34,$28,$68
lqd       $24,6192($10)

fma       $69,$35,$25,$69
shufb     $29,$16,$16,$12
fma       $70,$35,$26,$70
shufb     $30,$17,$17,$12
fma       $71,$35,$27,$71
shufb     $31,$18,$18,$12
fma       $72,$35,$28,$72
shufb     $32,$19,$19,$12

fma       $73,$36,$25,$73
fma       $74,$36,$26,$74
fma       $75,$36,$27,$75
```

```
          fma       $76,$36,$28,$76


#15
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,6400($10)
          fma       $66,$30,$22,$66
          lqd       $26,6416($10)
          fma       $67,$30,$23,$67
          lqd       $27,6432($10)
          fma       $68,$30,$24,$68
          lqd       $28,6448($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,6656($10)
          fma       $66,$34,$26,$66
          lqd       $22,6672($10)
          fma       $67,$34,$27,$67
          lqd       $23,6688($10)
          fma       $68,$34,$28,$68
          lqd       $24,6704($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,6912($10)
          fma       $62,$29,$22,$62
          lqd       $26,6928($10)
          fma       $63,$29,$23,$63
          lqd       $27,6944($10)
          fma       $64,$29,$24,$64
          lqd       $28,6960($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,240($9)
          fma       $62,$33,$26,$62
          lqd       $17,496($9)
          fma       $63,$33,$27,$63
          lqd       $18,752($9)
          fma       $64,$33,$28,$64
          lqd       $19,1008($9)

          fma       $65,$34,$25,$65
          lqd       $21,7168($10)
          fma       $66,$34,$26,$66
```

```
        lqd     $22,7184($10)
        fma     $67,$34,$27,$67
        lqd     $23,7200($10)
        fma     $68,$34,$28,$68
        lqd     $24,7216($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76


#16
        fma     $61,$29,$21,$61
        fma     $62,$29,$22,$62
        fma     $63,$29,$23,$63
        fma     $64,$29,$24,$64

        fma     $65,$30,$21,$65
        lqd     $25,7424($10)
        fma     $66,$30,$22,$66
        lqd     $26,7440($10)
        fma     $67,$30,$23,$67
        lqd     $27,7456($10)
        fma     $68,$30,$24,$68
        lqd     $28,7472($10)

        fma     $69,$31,$21,$69
        shufb   $33,$16,$16,$13
        fma     $70,$31,$22,$70
        shufb   $34,$17,$17,$13
        fma     $71,$31,$23,$71
        shufb   $35,$18,$18,$13
        fma     $72,$31,$24,$72
        shufb   $36,$19,$19,$13

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76

        fma     $61,$33,$25,$61
        fma     $62,$33,$26,$62
        fma     $63,$33,$27,$63
        fma     $64,$33,$28,$64

        fma     $65,$34,$25,$65
        lqd     $21,7680($10)
        fma     $66,$34,$26,$66
        lqd     $22,7696($10)
        fma     $67,$34,$27,$67
        lqd     $23,7712($10)
        fma     $68,$34,$28,$68
        lqd     $24,7728($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$14
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$14
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$14
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$14

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76


        fma     $61,$29,$21,$61
        lqd     $25,7936($10)
        fma     $62,$29,$22,$62
        lqd     $26,7952($10)
        fma     $63,$29,$23,$63
        lqd     $27,7968($10)
        fma     $64,$29,$24,$64
        lqd     $28,7984($10)

        fma     $65,$30,$21,$65
        shufb   $33,$16,$16,$15
        fma     $66,$30,$22,$66
        shufb   $34,$17,$17,$15
        fma     $67,$30,$23,$67
        shufb   $35,$18,$18,$15
        fma     $68,$30,$24,$68
        shufb   $36,$19,$19,$15

        fma     $69,$31,$21,$69
        rotqbyi $9,$59,0
        fma     $70,$31,$22,$70
#       ai      $10,$4,128
        shufb   $10,$55,$55,$14
        fma     $71,$31,$23,$71
        fma     $72,$31,$24,$72

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76


        fma     $61,$33,$25,$61
        lqd     $16,0($9)
        fma     $62,$33,$26,$62
        lqd     $17,256($9)
        fma     $63,$33,$27,$63
        lqd     $18,512($9)
        fma     $64,$33,$28,$64
        lqd     $19,768($9)

        fma     $65,$34,$25,$65
        lqd     $21,0($10)
        fma     $66,$34,$26,$66
        lqd     $22,16($10)
        fma     $67,$34,$27,$67
        lqd     $23,32($10)
        fma     $68,$34,$28,$68
        lqd     $24,48($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        stqd    $61,0($77)
        fma     $74,$36,$26,$74
        stqd    $62,16($77)
        fma     $75,$36,$27,$75
        stqd    $63,32($77)
        fma     $76,$36,$28,$76
        stqd    $64,48($77)



#N32-N47


#1
        fma     $37,$29,$21,$37
        lqd     $43,288($11)
        fma     $38,$29,$22,$38
        lqd     $44,304($11)
        fma     $39,$29,$23,$39
        lqd     $45,512($11)
        fma     $40,$29,$24,$40
        lqd     $46,528($11)

        fma     $41,$30,$21,$41
        lqd     $47,544($11)
        fma     $42,$30,$22,$42
        lqd     $48,560($11)
        fma     $43,$30,$23,$43
        lqd     $49,768($11)
        fma     $44,$30,$24,$44
        lqd     $50,784($11)

        fma     $45,$31,$21,$45
        lqd     $51,800($11)
        fma     $46,$31,$22,$46
        lqd     $52,816($11)
        fma     $47,$31,$23,$47
        lqd     $25,256($10)
        fma     $48,$31,$24,$48
        lqd     $26,272($10)

        fma     $49,$32,$21,$49
        lqd     $27,288($10)
        fma     $50,$32,$22,$50
        shufb   $33,$16,$16,$13
        fma     $51,$32,$23,$51
        lqd     $28,304($10)
        fma     $52,$32,$24,$52
        shufb   $34,$17,$17,$13

        fma     $37,$33,$25,$37
        fma     $38,$33,$26,$38
        fma     $39,$33,$27,$39
        shufb   $35,$18,$18,$13
        fma     $40,$33,$28,$40
        shufb   $36,$19,$19,$13

        fma     $41,$34,$25,$41
        lqd     $21,512($10)
        fma     $42,$34,$26,$42
        lqd     $22,528($10)
        fma     $43,$34,$27,$43
        lqd     $23,544($10)
        fma     $44,$34,$28,$44
        lqd     $24,560($10)

        fma     $45,$35,$25,$45
```

```
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,768($10)
          fma       $38,$29,$22,$38
          lqd       $26,784($10)
          fma       $39,$29,$23,$39
          lqd       $27,800($10)
          fma       $40,$29,$24,$40
          lqd       $28,816($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,16($9)
          fma       $38,$33,$26,$38
          lqd       $17,272($9)
          fma       $39,$33,$27,$39
          lqd       $18,528($9)
          fma       $40,$33,$28,$40
          lqd       $19,784($9)

          fma       $41,$34,$25,$41
          lqd       $21,1024($10)
          fma       $42,$34,$26,$42
          lqd       $22,1040($10)
          fma       $43,$34,$27,$43
          lqd       $23,1056($10)
          fma       $44,$34,$28,$44
          lqd       $24,1072($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#2
          fma       $37,$29,$21,$37
          stqd      $65,256($77)
          fma       $38,$29,$22,$38
          stqd      $66,272($77)
          fma       $39,$29,$23,$39
          stqd      $67,288($77)
          fma       $40,$29,$24,$40
          stqd      $68,304($77)

          fma       $41,$30,$21,$41
          lqd       $25,1280($10)
          fma       $42,$30,$22,$42
          lqd       $26,1296($10)
          fma       $43,$30,$23,$43
          lqd       $27,1312($10)
          fma       $44,$30,$24,$44
          lqd       $28,1328($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
```

```
shufb     $36,$19,$19,$13

fma       $49,$32,$21,$49
stqd      $69,512($77)
fma       $50,$32,$22,$50
stqd      $70,528($77)
fma       $51,$32,$23,$51
stqd      $71,544($77)
fma       $52,$32,$24,$52
stqd      $72,560($77)

fma       $37,$33,$25,$37
stqd      $73,768($77)
fma       $38,$33,$26,$38
stqd      $74,784($77)
fma       $39,$33,$27,$39
stqd      $75,800($77)
fma       $40,$33,$28,$40
stqd      $76,816($77)
ai        $77,$77,128
lnop

fma       $41,$34,$25,$41
lqd       $21,1536($10)
fma       $42,$34,$26,$42
lqd       $22,1552($10)
fma       $43,$34,$27,$43
lqd       $23,1568($10)
fma       $44,$34,$28,$44
lqd       $24,1584($10)

fma       $45,$35,$25,$45
shufb     $29,$16,$16,$14
fma       $46,$35,$26,$46
shufb     $30,$17,$17,$14
fma       $47,$35,$27,$47
shufb     $31,$18,$18,$14
fma       $48,$35,$28,$48
shufb     $32,$19,$19,$14

fma       $49,$36,$25,$49
fma       $50,$36,$26,$50
fma       $51,$36,$27,$51
fma       $52,$36,$28,$52


fma       $37,$29,$21,$37
lqd       $25,1792($10)
fma       $38,$29,$22,$38
lqd       $26,1808($10)
fma       $39,$29,$23,$39
lqd       $27,1824($10)
fma       $40,$29,$24,$40
lqd       $28,1840($10)

fma       $41,$30,$21,$41
shufb     $33,$16,$16,$15
fma       $42,$30,$22,$42
shufb     $34,$17,$17,$15
fma       $43,$30,$23,$43
shufb     $35,$18,$18,$15
fma       $44,$30,$24,$44
shufb     $36,$19,$19,$15

fma       $45,$31,$21,$45
fma       $46,$31,$22,$46
fma       $47,$31,$23,$47
fma       $48,$31,$24,$48

fma       $49,$32,$21,$49
fma       $50,$32,$22,$50
fma       $51,$32,$23,$51
fma       $52,$32,$24,$52


fma       $37,$33,$25,$37
lqd       $16,32($9)
fma       $38,$33,$26,$38
lqd       $17,288($9)
fma       $39,$33,$27,$39
lqd       $18,544($9)
fma       $40,$33,$28,$40
lqd       $19,800($9)

fma       $41,$34,$25,$41
lqd       $21,2048($10)
fma       $42,$34,$26,$42
lqd       $22,2064($10)
fma       $43,$34,$27,$43
lqd       $23,2080($10)
fma       $44,$34,$28,$44
lqd       $24,2096($10)

fma       $45,$35,$25,$45
shufb     $29,$16,$16,$12
fma       $46,$35,$26,$46
shufb     $30,$17,$17,$12
fma       $47,$35,$27,$47
shufb     $31,$18,$18,$12
fma       $48,$35,$28,$48
shufb     $32,$19,$19,$12

fma       $49,$36,$25,$49
```

```
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#3
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,2304($10)
          fma      $42,$30,$22,$42
          lqd      $26,2320($10)
          fma      $43,$30,$23,$43
          lqd      $27,2336($10)
          fma      $44,$30,$24,$44
          lqd      $28,2352($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          lqd      $61,0($77)
          fma      $50,$32,$22,$50
          lqd      $62,16($77)
          fma      $51,$32,$23,$51
          lqd      $63,32($77)
          fma      $52,$32,$24,$52
          lqd      $64,48($77)

          fma      $37,$33,$25,$37
          lqd      $65,256($77)
          fma      $38,$33,$26,$38
          lqd      $66,272($77)
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,2560($10)
          fma      $42,$34,$26,$42
          lqd      $22,2576($10)
          fma      $43,$34,$27,$43
          lqd      $23,2592($10)
          fma      $44,$34,$28,$44
          lqd      $24,2608($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,2816($10)
          fma      $38,$29,$22,$38
          lqd      $26,2832($10)
          fma      $39,$29,$23,$39
          lqd      $27,2848($10)
          fma      $40,$29,$24,$40
          lqd      $28,2864($10)

          fma      $41,$30,$21,$41
          shufb    $33,$16,$16,$15
          fma      $42,$30,$22,$42
          shufb    $34,$17,$17,$15
          fma      $43,$30,$23,$43
          shufb    $35,$18,$18,$15
          fma      $44,$30,$24,$44
          shufb    $36,$19,$19,$15

          fma      $45,$31,$21,$45
          fma      $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,48($9)
          fma      $38,$33,$26,$38
          lqd      $17,304($9)
          fma       $39,$33,$27,$39
          lqd       $18,560($9)
          fma       $40,$33,$28,$40
          lqd       $19,816($9)

          fma       $41,$34,$25,$41
          lqd       $21,3072($10)
          fma       $42,$34,$26,$42
          lqd       $22,3088($10)
          fma       $43,$34,$27,$43
          lqd       $23,3104($10)
          fma       $44,$34,$28,$44
          lqd       $24,3120($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#4
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,3328($10)
          fma       $42,$30,$22,$42
          lqd       $26,3344($10)
          fma       $43,$30,$23,$43
          lqd       $27,3360($10)
          fma       $44,$30,$24,$44
          lqd       $28,3376($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,3584($10)
          fma       $42,$34,$26,$42
          lqd       $22,3600($10)
          fma       $43,$34,$27,$43
          lqd       $23,3616($10)
          fma       $44,$34,$28,$44
          lqd       $24,3632($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,3840($10)
          fma       $38,$29,$22,$38
          lqd       $26,3856($10)
          fma       $39,$29,$23,$39
          lqd       $27,3872($10)
          fma       $40,$29,$24,$40
          lqd       $28,3888($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
```

```
            fma      $44,$30,$24,$44
            shufb    $36,$19,$19,$15

            fma      $45,$31,$21,$45
            fma      $46,$31,$22,$46
            fma      $47,$31,$23,$47
            fma      $48,$31,$24,$48

            fma      $49,$32,$21,$49
            fma      $50,$32,$22,$50
            fma      $51,$32,$23,$51
            fma      $52,$32,$24,$52


            fma      $37,$33,$25,$37
            lqd      $16,64($9)
            fma      $38,$33,$26,$38
            lqd      $17,320($9)
            fma      $39,$33,$27,$39
            lqd      $18,576($9)
            fma      $40,$33,$28,$40
            lqd      $19,832($9)

            fma      $41,$34,$25,$41
            lqd      $21,4096($10)
            fma      $42,$34,$26,$42
            lqd      $22,4112($10)
            fma      $43,$34,$27,$43
            lqd      $23,4128($10)
            fma      $44,$34,$28,$44
            lqd      $24,4144($10)

            fma      $45,$35,$25,$45
            shufb    $29,$16,$16,$12
            fma      $46,$35,$26,$46
            shufb    $30,$17,$17,$12
            fma      $47,$35,$27,$47
            shufb    $31,$18,$18,$12
            fma      $48,$35,$28,$48
            shufb    $32,$19,$19,$12

            fma      $49,$36,$25,$49
            fma      $50,$36,$26,$50
            fma      $51,$36,$27,$51
            fma      $52,$36,$28,$52



#5
            fma      $37,$29,$21,$37
            fma      $38,$29,$22,$38
            fma      $39,$29,$23,$39
            fma      $40,$29,$24,$40

            fma      $41,$30,$21,$41
            lqd      $25,4352($10)
            fma      $42,$30,$22,$42
            lqd      $26,4368($10)
            fma      $43,$30,$23,$43
            lqd      $27,4384($10)
            fma      $44,$30,$24,$44
            lqd      $28,4400($10)

            fma      $45,$31,$21,$45
            shufb    $33,$16,$16,$13
            fma      $46,$31,$22,$46
            shufb    $34,$17,$17,$13
            fma      $47,$31,$23,$47
            shufb    $35,$18,$18,$13
            fma      $48,$31,$24,$48
            shufb    $36,$19,$19,$13

            fma      $49,$32,$21,$49
            fma      $50,$32,$22,$50
            fma      $51,$32,$23,$51
            fma      $52,$32,$24,$52

            fma      $37,$33,$25,$37
            fma      $38,$33,$26,$38
            fma      $39,$33,$27,$39
            fma      $40,$33,$28,$40

            fma      $41,$34,$25,$41
            lqd      $21,4608($10)
            fma      $42,$34,$26,$42
            lqd      $22,4624($10)
            fma      $43,$34,$27,$43
            lqd      $23,4640($10)
            fma      $44,$34,$28,$44
            lqd      $24,4656($10)

            fma      $45,$35,$25,$45
            shufb    $29,$16,$16,$14
            fma      $46,$35,$26,$46
            shufb    $30,$17,$17,$14
            fma      $47,$35,$27,$47
            shufb    $31,$18,$18,$14
            fma      $48,$35,$28,$48
            shufb    $32,$19,$19,$14

            fma      $49,$36,$25,$49
            fma      $50,$36,$26,$50
            fma      $51,$36,$27,$51
```

```
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,4864($10)
          fma       $38,$29,$22,$38
          lqd       $26,4880($10)
          fma       $39,$29,$23,$39
          lqd       $27,4896($10)
          fma       $40,$29,$24,$40
          lqd       $28,4912($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,80($9)
          fma       $38,$33,$26,$38
          lqd       $17,336($9)
          fma       $39,$33,$27,$39
          lqd       $18,592($9)
          fma       $40,$33,$28,$40
          lqd       $19,848($9)

          fma       $41,$34,$25,$41
          lqd       $21,5120($10)
          fma       $42,$34,$26,$42
          lqd       $22,5136($10)
          fma       $43,$34,$27,$43
          lqd       $23,5152($10)
          fma       $44,$34,$28,$44
          lqd       $24,5168($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#6
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,5376($10)
          fma       $42,$30,$22,$42
          lqd       $26,5392($10)
          fma       $43,$30,$23,$43
          lqd       $27,5408($10)
          fma       $44,$30,$24,$44
          lqd       $28,5424($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,5632($10)
          fma       $42,$34,$26,$42
```

```
        lqd      $22,5648($10)
        fma      $43,$34,$27,$43
        lqd      $23,5664($10)
        fma      $44,$34,$28,$44
        lqd      $24,5680($10)

        fma      $45,$35,$25,$45
        shufb    $29,$16,$16,$14
        fma      $46,$35,$26,$46
        shufb    $30,$17,$17,$14
        fma      $47,$35,$27,$47
        shufb    $31,$18,$18,$14
        fma      $48,$35,$28,$48
        shufb    $32,$19,$19,$14

        fma      $49,$36,$25,$49
        fma      $50,$36,$26,$50
        fma      $51,$36,$27,$51
        fma      $52,$36,$28,$52


        fma      $37,$29,$21,$37
        lqd      $25,5888($10)
        fma      $38,$29,$22,$38
        lqd      $26,5904($10)
        fma      $39,$29,$23,$39
        lqd      $27,5920($10)
        fma      $40,$29,$24,$40
        lqd      $28,5936($10)

        fma      $41,$30,$21,$41
        shufb    $33,$16,$16,$15
        fma      $42,$30,$22,$42
        shufb    $34,$17,$17,$15
        fma      $43,$30,$23,$43
        shufb    $35,$18,$18,$15
        fma      $44,$30,$24,$44
        shufb    $36,$19,$19,$15

        fma      $45,$31,$21,$45
        fma      $46,$31,$22,$46
        fma      $47,$31,$23,$47
        fma      $48,$31,$24,$48

        fma      $49,$32,$21,$49
        fma      $50,$32,$22,$50
        fma      $51,$32,$23,$51
        fma      $52,$32,$24,$52


        fma      $37,$33,$25,$37
        lqd      $16,96($9)
        fma      $38,$33,$26,$38
        lqd      $17,352($9)
        fma      $39,$33,$27,$39
        lqd      $18,608($9)
        fma      $40,$33,$28,$40
        lqd      $19,864($9)

        fma      $41,$34,$25,$41
        lqd      $21,6144($10)
        fma      $42,$34,$26,$42
        lqd      $22,6160($10)
        fma      $43,$34,$27,$43
        lqd      $23,6176($10)
        fma      $44,$34,$28,$44
        lqd      $24,6192($10)

        fma      $45,$35,$25,$45
        shufb    $29,$16,$16,$12
        fma      $46,$35,$26,$46
        shufb    $30,$17,$17,$12
        fma      $47,$35,$27,$47
        shufb    $31,$18,$18,$12
        fma      $48,$35,$28,$48
        shufb    $32,$19,$19,$12

        fma      $49,$36,$25,$49
        fma      $50,$36,$26,$50
        fma      $51,$36,$27,$51
        fma      $52,$36,$28,$52



#7
        fma      $37,$29,$21,$37
        fma      $38,$29,$22,$38
        fma      $39,$29,$23,$39
        fma      $40,$29,$24,$40

        fma      $41,$30,$21,$41
        lqd      $25,6400($10)
        fma      $42,$30,$22,$42
        lqd      $26,6416($10)
        fma      $43,$30,$23,$43
        lqd      $27,6432($10)
        fma      $44,$30,$24,$44
        lqd      $28,6448($10)

        fma      $45,$31,$21,$45
        shufb    $33,$16,$16,$13
        fma      $46,$31,$22,$46
        shufb    $34,$17,$17,$13
        fma     $47,$31,$23,$47
        shufb   $35,$18,$18,$13
        fma     $48,$31,$24,$48
        shufb   $36,$19,$19,$13

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52

        fma     $37,$33,$25,$37
        fma     $38,$33,$26,$38
        fma     $39,$33,$27,$39
        fma     $40,$33,$28,$40

        fma     $41,$34,$25,$41
        lqd     $21,6656($10)
        fma     $42,$34,$26,$42
        lqd     $22,6672($10)
        fma     $43,$34,$27,$43
        lqd     $23,6688($10)
        fma     $44,$34,$28,$44
        lqd     $24,6704($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$14
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$14
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$14
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$14

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52


        fma     $37,$29,$21,$37
        lqd     $25,6912($10)
        fma     $38,$29,$22,$38
        lqd     $26,6928($10)
        fma     $39,$29,$23,$39
        lqd     $27,6944($10)
        fma     $40,$29,$24,$40
        lqd     $28,6960($10)

        fma     $41,$30,$21,$41
        shufb   $33,$16,$16,$15
        fma     $42,$30,$22,$42
        shufb   $34,$17,$17,$15
        fma     $43,$30,$23,$43
        shufb   $35,$18,$18,$15
        fma     $44,$30,$24,$44
        shufb   $36,$19,$19,$15

        fma     $45,$31,$21,$45
        fma     $46,$31,$22,$46
        fma     $47,$31,$23,$47
        fma     $48,$31,$24,$48

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52


        fma     $37,$33,$25,$37
        lqd     $16,112($9)
        fma     $38,$33,$26,$38
        lqd     $17,368($9)
        fma     $39,$33,$27,$39
        lqd     $18,624($9)
        fma     $40,$33,$28,$40
        lqd     $19,880($9)

        fma     $41,$34,$25,$41
        lqd     $21,7168($10)
        fma     $42,$34,$26,$42
        lqd     $22,7184($10)
        fma     $43,$34,$27,$43
        lqd     $23,7200($10)
        fma     $44,$34,$28,$44
        lqd     $24,7216($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$12
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$12
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$12
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$12

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52


#8
```

```
        fma     $37,$29,$21,$37
        fma     $38,$29,$22,$38
        fma     $39,$29,$23,$39
        fma     $40,$29,$24,$40

        fma     $41,$30,$21,$41
        lqd     $25,7424($10)
        fma     $42,$30,$22,$42
        lqd     $26,7440($10)
        fma     $43,$30,$23,$43
        lqd     $27,7456($10)
        fma     $44,$30,$24,$44
        lqd     $28,7472($10)

        fma     $45,$31,$21,$45
        shufb   $33,$16,$16,$13
        fma     $46,$31,$22,$46
        shufb   $34,$17,$17,$13
        fma     $47,$31,$23,$47
        shufb   $35,$18,$18,$13
        fma     $48,$31,$24,$48
        shufb   $36,$19,$19,$13

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52

        fma     $37,$33,$25,$37
        fma     $38,$33,$26,$38
        fma     $39,$33,$27,$39
        fma     $40,$33,$28,$40

        fma     $41,$34,$25,$41
        lqd     $21,7680($10)
        fma     $42,$34,$26,$42
        lqd     $22,7696($10)
        fma     $43,$34,$27,$43
        lqd     $23,7712($10)
        fma     $44,$34,$28,$44
        lqd     $24,7728($10)

        fma     $45,$35,$25,$45
        shufb   $29,$16,$16,$14
        fma     $46,$35,$26,$46
        shufb   $30,$17,$17,$14
        fma     $47,$35,$27,$47
        shufb   $31,$18,$18,$14
        fma     $48,$35,$28,$48
        shufb   $32,$19,$19,$14

        fma     $49,$36,$25,$49
        fma     $50,$36,$26,$50
        fma     $51,$36,$27,$51
        fma     $52,$36,$28,$52


        fma     $37,$29,$21,$37
        lqd     $25,7936($10)
        fma     $38,$29,$22,$38
        lqd     $26,7952($10)
        fma     $39,$29,$23,$39
        lqd     $27,7968($10)
        fma     $40,$29,$24,$40
        lqd     $28,7984($10)

        fma     $41,$30,$21,$41
        shufb   $33,$16,$16,$15
        fma     $42,$30,$22,$42
        shufb   $34,$17,$17,$15
        fma     $43,$30,$23,$43
        shufb   $35,$18,$18,$15
        fma     $44,$30,$24,$44
        shufb   $36,$19,$19,$15

        fma     $45,$31,$21,$45
        fma     $46,$31,$22,$46
        fma     $47,$31,$23,$47
#       a       $10,$10,$78
        shufb   $10,$58,$58,$14
        fma     $48,$31,$24,$48
        lnop

        fma     $49,$32,$21,$49
        fma     $50,$32,$22,$50
        fma     $51,$32,$23,$51
        fma     $52,$32,$24,$52


        fma     $37,$33,$25,$37
        lqd     $16,128($9)
        fma     $38,$33,$26,$38
        lqd     $17,384($9)
        fma     $39,$33,$27,$39
        lqd     $18,640($9)
        fma     $40,$33,$28,$40
        lqd     $19,896($9)

        fma     $41,$34,$25,$41
        lqd     $21,0($10)
        fma     $42,$34,$26,$42
        lqd     $22,16($10)
        fma     $43,$34,$27,$43
```

```
          lqd       $23,32($10)
          fma       $44,$34,$28,$44
          lqd       $24,48($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52



#9
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,256($10)
          fma       $42,$30,$22,$42
          lqd       $26,272($10)
          fma       $43,$30,$23,$43
          lqd       $27,288($10)
          fma       $44,$30,$24,$44
          lqd       $28,304($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,512($10)
          fma       $42,$34,$26,$42
          lqd       $22,528($10)
          fma       $43,$34,$27,$43
          lqd       $23,544($10)
          fma       $44,$34,$28,$44
          lqd       $24,560($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,768($10)
          fma       $38,$29,$22,$38
          lqd       $26,784($10)
          fma       $39,$29,$23,$39
          lqd       $27,800($10)
          fma       $40,$29,$24,$40
          lqd       $28,816($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,144($9)
          fma      $38,$33,$26,$38
          lqd      $17,400($9)
          fma      $39,$33,$27,$39
          lqd      $18,656($9)
          fma      $40,$33,$28,$40
          lqd      $19,912($9)

          fma      $41,$34,$25,$41
          lqd      $21,1024($10)
          fma      $42,$34,$26,$42
          lqd      $22,1040($10)
          fma      $43,$34,$27,$43
          lqd      $23,1056($10)
          fma      $44,$34,$28,$44
          lqd      $24,1072($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$12
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$12
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$12
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$12

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#10
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,1280($10)
          fma      $42,$30,$22,$42
          lqd      $26,1296($10)
          fma      $43,$30,$23,$43
          lqd      $27,1312($10)
          fma      $44,$30,$24,$44
          lqd      $28,1328($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,1536($10)
          fma      $42,$34,$26,$42
          lqd      $22,1552($10)
          fma      $43,$34,$27,$43
          lqd      $23,1568($10)
          fma      $44,$34,$28,$44
          lqd      $24,1584($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,1792($10)
          fma      $38,$29,$22,$38
          lqd      $26,1808($10)
          fma      $39,$29,$23,$39
          lqd      $27,1824($10)
          fma      $40,$29,$24,$40
```

```
            lqd        $28,1840($10)

            fma        $41,$30,$21,$41
            shufb      $33,$16,$16,$15
            fma        $42,$30,$22,$42
            shufb      $34,$17,$17,$15
            fma        $43,$30,$23,$43
            shufb      $35,$18,$18,$15
            fma        $44,$30,$24,$44
            shufb      $36,$19,$19,$15

            fma        $45,$31,$21,$45
            fma        $46,$31,$22,$46
            fma        $47,$31,$23,$47
            fma        $48,$31,$24,$48

            fma        $49,$32,$21,$49
            fma        $50,$32,$22,$50
            fma        $51,$32,$23,$51
            fma        $52,$32,$24,$52


            fma        $37,$33,$25,$37
            lqd        $16,160($9)
            fma        $38,$33,$26,$38
            lqd        $17,416($9)
            fma        $39,$33,$27,$39
            lqd        $18,672($9)
            fma        $40,$33,$28,$40
            lqd        $19,928($9)

            fma        $41,$34,$25,$41
            lqd        $21,2048($10)
            fma        $42,$34,$26,$42
            lqd        $22,2064($10)
            fma        $43,$34,$27,$43
            lqd        $23,2080($10)
            fma        $44,$34,$28,$44
            lqd        $24,2096($10)

            fma        $45,$35,$25,$45
            shufb      $29,$16,$16,$12
            fma        $46,$35,$26,$46
            shufb      $30,$17,$17,$12
            fma        $47,$35,$27,$47
            shufb      $31,$18,$18,$12
            fma        $48,$35,$28,$48
            shufb      $32,$19,$19,$12

            fma        $49,$36,$25,$49
            fma        $50,$36,$26,$50
            fma        $51,$36,$27,$51
            fma        $52,$36,$28,$52



#11
            fma        $37,$29,$21,$37
            fma        $38,$29,$22,$38
            fma        $39,$29,$23,$39
            fma        $40,$29,$24,$40

            fma        $41,$30,$21,$41
            lqd        $25,2304($10)
            fma        $42,$30,$22,$42
            lqd        $26,2320($10)
            fma        $43,$30,$23,$43
            lqd        $27,2336($10)
            fma        $44,$30,$24,$44
            lqd        $28,2352($10)

            fma        $45,$31,$21,$45
            shufb      $33,$16,$16,$13
            fma        $46,$31,$22,$46
            shufb      $34,$17,$17,$13
            fma        $47,$31,$23,$47
            shufb      $35,$18,$18,$13
            fma        $48,$31,$24,$48
            shufb      $36,$19,$19,$13

            fma        $49,$32,$21,$49
            fma        $50,$32,$22,$50
            fma        $51,$32,$23,$51
            fma        $52,$32,$24,$52

            fma        $37,$33,$25,$37
            fma        $38,$33,$26,$38
            fma        $39,$33,$27,$39
            fma        $40,$33,$28,$40

            fma        $41,$34,$25,$41
            lqd        $21,2560($10)
            fma        $42,$34,$26,$42
            lqd        $22,2576($10)
            fma        $43,$34,$27,$43
            lqd        $23,2592($10)
            fma        $44,$34,$28,$44
            lqd        $24,2608($10)

            fma        $45,$35,$25,$45
            shufb      $29,$16,$16,$14
            fma        $46,$35,$26,$46
            shufb      $30,$17,$17,$14
            fma       $47,$35,$27,$47
            shufb     $31,$18,$18,$14
            fma       $48,$35,$28,$48
            shufb     $32,$19,$19,$14

            fma       $49,$36,$25,$49
            fma       $50,$36,$26,$50
            fma       $51,$36,$27,$51
            fma       $52,$36,$28,$52


            fma       $37,$29,$21,$37
            lqd       $25,2816($10)
            fma       $38,$29,$22,$38
            lqd       $26,2832($10)
            fma       $39,$29,$23,$39
            lqd       $27,2848($10)
            fma       $40,$29,$24,$40
            lqd       $28,2864($10)

            fma       $41,$30,$21,$41
            shufb     $33,$16,$16,$15
            fma       $42,$30,$22,$42
            shufb     $34,$17,$17,$15
            fma       $43,$30,$23,$43
            shufb     $35,$18,$18,$15
            fma       $44,$30,$24,$44
            shufb     $36,$19,$19,$15

            fma       $45,$31,$21,$45
            fma       $46,$31,$22,$46
            fma       $47,$31,$23,$47
            fma       $48,$31,$24,$48

            fma       $49,$32,$21,$49
            fma       $50,$32,$22,$50
            fma       $51,$32,$23,$51
            fma       $52,$32,$24,$52


            fma       $37,$33,$25,$37
            lqd       $16,176($9)
            fma       $38,$33,$26,$38
            lqd       $17,432($9)
            fma       $39,$33,$27,$39
            lqd       $18,688($9)
            fma       $40,$33,$28,$40
            lqd       $19,944($9)

            fma       $41,$34,$25,$41
            lqd       $21,3072($10)
            fma       $42,$34,$26,$42
            lqd       $22,3088($10)
            fma       $43,$34,$27,$43
            lqd       $23,3104($10)
            fma       $44,$34,$28,$44
            lqd       $24,3120($10)

            fma       $45,$35,$25,$45
            shufb     $29,$16,$16,$12
            fma       $46,$35,$26,$46
            shufb     $30,$17,$17,$12
            fma       $47,$35,$27,$47
            shufb     $31,$18,$18,$12
            fma       $48,$35,$28,$48
            shufb     $32,$19,$19,$12

            fma       $49,$36,$25,$49
            fma       $50,$36,$26,$50
            fma       $51,$36,$27,$51
            fma       $52,$36,$28,$52



    #12
            fma       $37,$29,$21,$37
            fma       $38,$29,$22,$38
            fma       $39,$29,$23,$39
            fma       $40,$29,$24,$40

            fma       $41,$30,$21,$41
            lqd       $25,3328($10)
            fma       $42,$30,$22,$42
            lqd       $26,3344($10)
            fma       $43,$30,$23,$43
            lqd       $27,3360($10)
            fma       $44,$30,$24,$44
            lqd       $28,3376($10)

            fma       $45,$31,$21,$45
            shufb     $33,$16,$16,$13
            fma       $46,$31,$22,$46
            shufb     $34,$17,$17,$13
            fma       $47,$31,$23,$47
            shufb     $35,$18,$18,$13
            fma       $48,$31,$24,$48
            shufb     $36,$19,$19,$13

            fma       $49,$32,$21,$49
            fma       $50,$32,$22,$50
            fma       $51,$32,$23,$51
            fma       $52,$32,$24,$52
```

```
        fma      $37,$33,$25,$37
        fma      $38,$33,$26,$38
        fma      $39,$33,$27,$39
        fma      $40,$33,$28,$40

        fma      $41,$34,$25,$41
        lqd      $21,3584($10)
        fma      $42,$34,$26,$42
        lqd      $22,3600($10)
        fma      $43,$34,$27,$43
        lqd      $23,3616($10)
        fma      $44,$34,$28,$44
        lqd      $24,3632($10)

        fma      $45,$35,$25,$45
        shufb    $29,$16,$16,$14
        fma      $46,$35,$26,$46
        shufb    $30,$17,$17,$14
        fma      $47,$35,$27,$47
        shufb    $31,$18,$18,$14
        fma      $48,$35,$28,$48
        shufb    $32,$19,$19,$14

        fma      $49,$36,$25,$49
        fma      $50,$36,$26,$50
        fma      $51,$36,$27,$51
        fma      $52,$36,$28,$52


        fma      $37,$29,$21,$37
        lqd      $25,3840($10)
        fma      $38,$29,$22,$38
        lqd      $26,3856($10)
        fma      $39,$29,$23,$39
        lqd      $27,3872($10)
        fma      $40,$29,$24,$40
        lqd      $28,3888($10)

        fma      $41,$30,$21,$41
        shufb    $33,$16,$16,$15
        fma      $42,$30,$22,$42
        shufb    $34,$17,$17,$15
        fma      $43,$30,$23,$43
        shufb    $35,$18,$18,$15
        fma      $44,$30,$24,$44
        shufb    $36,$19,$19,$15

        fma      $45,$31,$21,$45
        fma      $46,$31,$22,$46
        fma      $47,$31,$23,$47
        fma      $48,$31,$24,$48

        fma      $49,$32,$21,$49
        fma      $50,$32,$22,$50
        fma      $51,$32,$23,$51
        fma      $52,$32,$24,$52


        fma      $37,$33,$25,$37
        lqd      $16,192($9)
        fma      $38,$33,$26,$38
        lqd      $17,448($9)
        fma      $39,$33,$27,$39
        lqd      $18,704($9)
        fma      $40,$33,$28,$40
        lqd      $19,960($9)

        fma      $41,$34,$25,$41
        lqd      $21,4096($10)
        fma      $42,$34,$26,$42
        lqd      $22,4112($10)
        fma      $43,$34,$27,$43
        lqd      $23,4128($10)
        fma      $44,$34,$28,$44
        lqd      $24,4144($10)

        fma      $45,$35,$25,$45
        shufb    $29,$16,$16,$12
        fma      $46,$35,$26,$46
        shufb    $30,$17,$17,$12
        fma      $47,$35,$27,$47
        shufb    $31,$18,$18,$12
        fma      $48,$35,$28,$48
        shufb    $32,$19,$19,$12

        fma      $49,$36,$25,$49
        fma      $50,$36,$26,$50
        fma      $51,$36,$27,$51
        fma      $52,$36,$28,$52



#13
        fma      $37,$29,$21,$37
        fma      $38,$29,$22,$38
        fma      $39,$29,$23,$39
        fma      $40,$29,$24,$40

        fma      $41,$30,$21,$41
        lqd      $25,4352($10)
        fma      $42,$30,$22,$42
        lqd      $26,4368($10)
        fma      $43,$30,$23,$43
```

```
lqd       $27,4384($10)
fma       $44,$30,$24,$44
lqd       $28,4400($10)

fma       $45,$31,$21,$45
shufb     $33,$16,$16,$13
fma       $46,$31,$22,$46
shufb     $34,$17,$17,$13
fma       $47,$31,$23,$47
shufb     $35,$18,$18,$13
fma       $48,$31,$24,$48
shufb     $36,$19,$19,$13

fma       $49,$32,$21,$49
fma       $50,$32,$22,$50
fma       $51,$32,$23,$51
fma       $52,$32,$24,$52

fma       $37,$33,$25,$37
fma       $38,$33,$26,$38
fma       $39,$33,$27,$39
fma       $40,$33,$28,$40

fma       $41,$34,$25,$41
lqd       $21,4608($10)
fma       $42,$34,$26,$42
lqd       $22,4624($10)
fma       $43,$34,$27,$43
lqd       $23,4640($10)
fma       $44,$34,$28,$44
lqd       $24,4656($10)

fma       $45,$35,$25,$45
shufb     $29,$16,$16,$14
fma       $46,$35,$26,$46
shufb     $30,$17,$17,$14
fma       $47,$35,$27,$47
shufb     $31,$18,$18,$14
fma       $48,$35,$28,$48
shufb     $32,$19,$19,$14

fma       $49,$36,$25,$49
fma       $50,$36,$26,$50
fma       $51,$36,$27,$51
fma       $52,$36,$28,$52


fma       $37,$29,$21,$37
lqd       $25,4864($10)
fma       $38,$29,$22,$38
lqd       $26,4880($10)
fma       $39,$29,$23,$39
lqd       $27,4896($10)
fma       $40,$29,$24,$40
lqd       $28,4912($10)

fma       $41,$30,$21,$41
shufb     $33,$16,$16,$15
fma       $42,$30,$22,$42
shufb     $34,$17,$17,$15
fma       $43,$30,$23,$43
shufb     $35,$18,$18,$15
fma       $44,$30,$24,$44
shufb     $36,$19,$19,$15

fma       $45,$31,$21,$45
fma       $46,$31,$22,$46
fma       $47,$31,$23,$47
fma       $48,$31,$24,$48

fma       $49,$32,$21,$49
fma       $50,$32,$22,$50
fma       $51,$32,$23,$51
fma       $52,$32,$24,$52


fma       $37,$33,$25,$37
lqd       $16,208($9)
fma       $38,$33,$26,$38
lqd       $17,464($9)
fma       $39,$33,$27,$39
lqd       $18,720($9)
fma       $40,$33,$28,$40
lqd       $19,976($9)

fma       $41,$34,$25,$41
lqd       $21,5120($10)
fma       $42,$34,$26,$42
lqd       $22,5136($10)
fma       $43,$34,$27,$43
lqd       $23,5152($10)
fma       $44,$34,$28,$44
lqd       $24,5168($10)

fma       $45,$35,$25,$45
shufb     $29,$16,$16,$12
fma       $46,$35,$26,$46
shufb     $30,$17,$17,$12
fma       $47,$35,$27,$47
shufb     $31,$18,$18,$12
fma       $48,$35,$28,$48
shufb     $32,$19,$19,$12
```

```
          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52



#14
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,5376($10)
          fma      $42,$30,$22,$42
          lqd      $26,5392($10)
          fma      $43,$30,$23,$43
          lqd      $27,5408($10)
          fma      $44,$30,$24,$44
          lqd      $28,5424($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,5632($10)
          fma      $42,$34,$26,$42
          lqd      $22,5648($10)
          fma      $43,$34,$27,$43
          lqd      $23,5664($10)
          fma      $44,$34,$28,$44
          lqd      $24,5680($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,5888($10)
          fma      $38,$29,$22,$38
          lqd      $26,5904($10)
          fma      $39,$29,$23,$39
          lqd      $27,5920($10)
          fma      $40,$29,$24,$40
          lqd      $28,5936($10)

          fma      $41,$30,$21,$41
          shufb    $33,$16,$16,$15
          fma      $42,$30,$22,$42
          shufb    $34,$17,$17,$15
          fma      $43,$30,$23,$43
          shufb    $35,$18,$18,$15
          fma      $44,$30,$24,$44
          shufb    $36,$19,$19,$15

          fma      $45,$31,$21,$45
          fma      $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,224($9)
          fma      $38,$33,$26,$38
          lqd      $17,480($9)
          fma      $39,$33,$27,$39
          lqd      $18,736($9)
          fma      $40,$33,$28,$40
          lqd      $19,992($9)
          fma       $41,$34,$25,$41
          lqd       $21,6144($10)
          fma       $42,$34,$26,$42
          lqd       $22,6160($10)
          fma       $43,$34,$27,$43
          lqd       $23,6176($10)
          fma       $44,$34,$28,$44
          lqd       $24,6192($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


#15
          fma       $37,$29,$21,$37
          fma       $38,$29,$22,$38
          fma       $39,$29,$23,$39
          fma       $40,$29,$24,$40

          fma       $41,$30,$21,$41
          lqd       $25,6400($10)
          fma       $42,$30,$22,$42
          lqd       $26,6416($10)
          fma       $43,$30,$23,$43
          lqd       $27,6432($10)
          fma       $44,$30,$24,$44
          lqd       $28,6448($10)

          fma       $45,$31,$21,$45
          shufb     $33,$16,$16,$13
          fma       $46,$31,$22,$46
          shufb     $34,$17,$17,$13
          fma       $47,$31,$23,$47
          shufb     $35,$18,$18,$13
          fma       $48,$31,$24,$48
          shufb     $36,$19,$19,$13

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52

          fma       $37,$33,$25,$37
          fma       $38,$33,$26,$38
          fma       $39,$33,$27,$39
          fma       $40,$33,$28,$40

          fma       $41,$34,$25,$41
          lqd       $21,6656($10)
          fma       $42,$34,$26,$42
          lqd       $22,6672($10)
          fma       $43,$34,$27,$43
          lqd       $23,6688($10)
          fma       $44,$34,$28,$44
          lqd       $24,6704($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$14
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$14
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$14
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$14

          fma       $49,$36,$25,$49
          fma       $50,$36,$26,$50
          fma       $51,$36,$27,$51
          fma       $52,$36,$28,$52


          fma       $37,$29,$21,$37
          lqd       $25,6912($10)
          fma       $38,$29,$22,$38
          lqd       $26,6928($10)
          fma       $39,$29,$23,$39
          lqd       $27,6944($10)
          fma       $40,$29,$24,$40
          lqd       $28,6960($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          fma       $46,$31,$22,$46
          fma      $47,$31,$23,$47
          fma      $48,$31,$24,$48

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52


          fma      $37,$33,$25,$37
          lqd      $16,240($9)
          fma      $38,$33,$26,$38
          lqd      $17,496($9)
          fma      $39,$33,$27,$39
          lqd      $18,752($9)
          fma      $40,$33,$28,$40
          lqd      $19,1008($9)

          fma      $41,$34,$25,$41
          lqd      $21,7168($10)
          fma      $42,$34,$26,$42
          lqd      $22,7184($10)
          fma      $43,$34,$27,$43
          lqd      $23,7200($10)
          fma      $44,$34,$28,$44
          lqd      $24,7216($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$12
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$12
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$12
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$12

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


#16
          fma      $37,$29,$21,$37
          fma      $38,$29,$22,$38
          fma      $39,$29,$23,$39
          fma      $40,$29,$24,$40

          fma      $41,$30,$21,$41
          lqd      $25,7424($10)
          fma      $42,$30,$22,$42
          lqd      $26,7440($10)
          fma      $43,$30,$23,$43
          lqd      $27,7456($10)
          fma      $44,$30,$24,$44
          lqd      $28,7472($10)

          fma      $45,$31,$21,$45
          shufb    $33,$16,$16,$13
          fma      $46,$31,$22,$46
          shufb    $34,$17,$17,$13
          fma      $47,$31,$23,$47
          shufb    $35,$18,$18,$13
          fma      $48,$31,$24,$48
          shufb    $36,$19,$19,$13

          fma      $49,$32,$21,$49
          fma      $50,$32,$22,$50
          fma      $51,$32,$23,$51
          fma      $52,$32,$24,$52

          fma      $37,$33,$25,$37
          fma      $38,$33,$26,$38
          fma      $39,$33,$27,$39
          fma      $40,$33,$28,$40

          fma      $41,$34,$25,$41
          lqd      $21,7680($10)
          fma      $42,$34,$26,$42
          lqd      $22,7696($10)
          fma      $43,$34,$27,$43
          lqd      $23,7712($10)
          fma      $44,$34,$28,$44
          lqd      $24,7728($10)

          fma      $45,$35,$25,$45
          shufb    $29,$16,$16,$14
          fma      $46,$35,$26,$46
          shufb    $30,$17,$17,$14
          fma      $47,$35,$27,$47
          shufb    $31,$18,$18,$14
          fma      $48,$35,$28,$48
          shufb    $32,$19,$19,$14

          fma      $49,$36,$25,$49
          fma      $50,$36,$26,$50
          fma      $51,$36,$27,$51
          fma      $52,$36,$28,$52


          fma      $37,$29,$21,$37
          lqd      $25,7936($10)
          fma      $38,$29,$22,$38
          lqd       $26,7952($10)
          fma       $39,$29,$23,$39
          lqd       $27,7968($10)
          fma       $40,$29,$24,$40
          lqd       $28,7984($10)

          fma       $41,$30,$21,$41
          shufb     $33,$16,$16,$15
          fma       $42,$30,$22,$42
          shufb     $34,$17,$17,$15
          fma       $43,$30,$23,$43
          shufb     $35,$18,$18,$15
          fma       $44,$30,$24,$44
          shufb     $36,$19,$19,$15

          fma       $45,$31,$21,$45
          rotqbyi   $9,$59,0
          fma       $46,$31,$22,$46
#         ai        $10,$4,192
          shufb     $10,$55,$55,$15
          fma       $47,$31,$23,$47
          fma       $48,$31,$24,$48

          fma       $49,$32,$21,$49
          fma       $50,$32,$22,$50
          fma       $51,$32,$23,$51
          fma       $52,$32,$24,$52


          fma       $37,$33,$25,$37
          lqd       $16,0($9)
          fma       $38,$33,$26,$38
          lqd       $17,256($9)
          fma       $39,$33,$27,$39
          lqd       $18,512($9)
          fma       $40,$33,$28,$40
          lqd       $19,768($9)

          fma       $41,$34,$25,$41
          lqd       $21,0($10)
          fma       $42,$34,$26,$42
          lqd       $22,16($10)
          fma       $43,$34,$27,$43
          lqd       $23,32($10)
          fma       $44,$34,$28,$44
          lqd       $24,48($10)

          fma       $45,$35,$25,$45
          shufb     $29,$16,$16,$12
          fma       $46,$35,$26,$46
          shufb     $30,$17,$17,$12
          fma       $47,$35,$27,$47
          shufb     $31,$18,$18,$12
          fma       $48,$35,$28,$48
          shufb     $32,$19,$19,$12

          fma       $49,$36,$25,$49
          stqd      $37,0($11)
          fma       $50,$36,$26,$50
          stqd      $38,16($11)
          fma       $51,$36,$27,$51
          stqd      $39,32($11)
          fma       $52,$36,$28,$52
          stqd      $40,48($11)



#N48-N63



#1
          fma       $61,$29,$21,$61
          lqd       $67,288($77)
          fma       $62,$29,$22,$62
          lqd       $68,304($77)
          fma       $63,$29,$23,$63
          lqd       $69,512($77)
          fma       $64,$29,$24,$64
          lqd       $70,528($77)

          fma       $65,$30,$21,$65
          lqd       $71,544($77)
          fma       $66,$30,$22,$66
          lqd       $72,560($77)
          fma       $67,$30,$23,$67
          lqd       $73,768($77)
          fma       $68,$30,$24,$68
          lqd       $74,784($77)

          fma       $69,$31,$21,$69
          lqd       $75,800($77)
          fma       $70,$31,$22,$70
          lqd       $76,816($77)
          fma       $71,$31,$23,$71
          lqd       $25,256($10)
          fma       $72,$31,$24,$72
          lqd       $26,272($10)

          fma       $73,$32,$21,$73
          lqd       $27,288($10)
          fma       $74,$32,$22,$74
          shufb     $33,$16,$16,$13
          fma       $75,$32,$23,$75
          lqd       $28,304($10)
          fma       $76,$32,$24,$76
          shufb     $34,$17,$17,$13

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          shufb     $35,$18,$18,$13
          fma       $64,$33,$28,$64
          shufb     $36,$19,$19,$13

          fma       $65,$34,$25,$65
          lqd       $21,512($10)
          fma       $66,$34,$26,$66
          lqd       $22,528($10)
          fma       $67,$34,$27,$67
          lqd       $23,544($10)
          fma       $68,$34,$28,$68
          lqd       $24,560($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,768($10)
          fma       $62,$29,$22,$62
          lqd       $26,784($10)
          fma       $63,$29,$23,$63
          lqd       $27,800($10)
          fma       $64,$29,$24,$64
          lqd       $28,816($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,16($9)
          fma       $62,$33,$26,$62
          lqd       $17,272($9)
          fma       $63,$33,$27,$63
          lqd       $18,528($9)
          fma       $64,$33,$28,$64
          lqd       $19,784($9)

          fma       $65,$34,$25,$65
          lqd       $21,1024($10)
          fma       $66,$34,$26,$66
          lqd       $22,1040($10)
          fma       $67,$34,$27,$67
          lqd       $23,1056($10)
          fma       $68,$34,$28,$68
          lqd       $24,1072($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#2
          fma       $61,$29,$21,$61
          stqd      $41,256($11)
```

```
fma       $62,$29,$22,$62
stqd      $42,272($11)
fma       $63,$29,$23,$63
stqd      $43,288($11)
fma       $64,$29,$24,$64
stqd      $44,304($11)

fma       $65,$30,$21,$65
lqd       $25,1280($10)
fma       $66,$30,$22,$66
lqd       $26,1296($10)
fma       $67,$30,$23,$67
lqd       $27,1312($10)
fma       $68,$30,$24,$68
lqd       $28,1328($10)

fma       $69,$31,$21,$69
shufb     $33,$16,$16,$13
fma       $70,$31,$22,$70
shufb     $34,$17,$17,$13
fma       $71,$31,$23,$71
shufb     $35,$18,$18,$13
fma       $72,$31,$24,$72
shufb     $36,$19,$19,$13

fma       $73,$32,$21,$73
stqd      $45,512($11)
fma       $74,$32,$22,$74
stqd      $46,528($11)
fma       $75,$32,$23,$75
stqd      $47,544($11)
fma       $76,$32,$24,$76
stqd      $48,560($11)

fma       $61,$33,$25,$61
stqd      $49,768($11)
fma       $62,$33,$26,$62
stqd      $50,784($11)
fma       $63,$33,$27,$63
stqd      $51,800($11)
fma       $64,$33,$28,$64
stqd      $52,816($11)
a         $11,$60,$56
lnop

fma       $65,$34,$25,$65
lqd       $21,1536($10)
fma       $66,$34,$26,$66
lqd       $22,1552($10)
fma       $67,$34,$27,$67
lqd       $23,1568($10)
fma       $68,$34,$28,$68
lqd       $24,1584($10)

fma       $69,$35,$25,$69
shufb     $29,$16,$16,$14
fma       $70,$35,$26,$70
shufb     $30,$17,$17,$14
fma       $71,$35,$27,$71
shufb     $31,$18,$18,$14
fma       $72,$35,$28,$72
shufb     $32,$19,$19,$14

fma       $73,$36,$25,$73
rotqbyi   $60,$11,0
fma       $74,$36,$26,$74
lnop
fma       $75,$36,$27,$75
fma       $76,$36,$28,$76


fma       $61,$29,$21,$61
lqd       $25,1792($10)
fma       $62,$29,$22,$62
lqd       $26,1808($10)
fma       $63,$29,$23,$63
lqd       $27,1824($10)
fma       $64,$29,$24,$64
lqd       $28,1840($10)

fma       $65,$30,$21,$65
shufb     $33,$16,$16,$15
fma       $66,$30,$22,$66
shufb     $34,$17,$17,$15
fma       $67,$30,$23,$67
shufb     $35,$18,$18,$15
fma       $68,$30,$24,$68
shufb     $36,$19,$19,$15

fma       $69,$31,$21,$69
fma       $70,$31,$22,$70
fma       $71,$31,$23,$71
fma       $72,$31,$24,$72

fma       $73,$32,$21,$73
fma       $74,$32,$22,$74
fma       $75,$32,$23,$75
fma       $76,$32,$24,$76


fma       $61,$33,$25,$61
lqd       $16,32($9)
fma       $62,$33,$26,$62
```

```
          lqd      $17,288($9)
          fma      $63,$33,$27,$63
          lqd      $18,544($9)
          fma      $64,$33,$28,$64
          lqd      $19,800($9)

          fma      $65,$34,$25,$65
          lqd      $21,2048($10)
          fma      $66,$34,$26,$66
          lqd      $22,2064($10)
          fma      $67,$34,$27,$67
          lqd      $23,2080($10)
          fma      $68,$34,$28,$68
          lqd      $24,2096($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$12
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$12
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$12
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$12

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76



#3
          fma      $61,$29,$21,$61
          fma      $62,$29,$22,$62
          fma      $63,$29,$23,$63
          fma      $64,$29,$24,$64

          fma      $65,$30,$21,$65
          lqd      $25,2304($10)
          fma      $66,$30,$22,$66
          lqd      $26,2320($10)
          fma      $67,$30,$23,$67
          lqd      $27,2336($10)
          fma      $68,$30,$24,$68
          lqd      $28,2352($10)

          fma      $69,$31,$21,$69
          shufb    $33,$16,$16,$13
          fma      $70,$31,$22,$70
          shufb    $34,$17,$17,$13
          fma      $71,$31,$23,$71
          shufb    $35,$18,$18,$13
          fma      $72,$31,$24,$72
          shufb    $36,$19,$19,$13

          fma      $73,$32,$21,$73
          lqd      $37,0($11)
          fma      $74,$32,$22,$74
          lqd      $38,16($11)
          fma      $75,$32,$23,$75
          lqd      $39,32($11)
          fma      $76,$32,$24,$76
          lqd      $40,48($11)

          fma      $61,$33,$25,$61
          lqd      $41,256($11)
          fma      $62,$33,$26,$62
          lqd      $42,272($11)
          fma      $63,$33,$27,$63
          fma      $64,$33,$28,$64

          fma      $65,$34,$25,$65
          lqd      $21,2560($10)
          fma      $66,$34,$26,$66
          lqd      $22,2576($10)
          fma      $67,$34,$27,$67
          lqd      $23,2592($10)
          fma      $68,$34,$28,$68
          lqd      $24,2608($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$14
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$14
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$14
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$14

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76


          fma      $61,$29,$21,$61
          lqd      $25,2816($10)
          fma      $62,$29,$22,$62
          lqd      $26,2832($10)
          fma      $63,$29,$23,$63
          lqd      $27,2848($10)
          fma      $64,$29,$24,$64
          lqd      $28,2864($10)
          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,48($9)
          fma       $62,$33,$26,$62
          lqd       $17,304($9)
          fma       $63,$33,$27,$63
          lqd       $18,560($9)
          fma       $64,$33,$28,$64
          lqd       $19,816($9)

          fma       $65,$34,$25,$65
          lqd       $21,3072($10)
          fma       $66,$34,$26,$66
          lqd       $22,3088($10)
          fma       $67,$34,$27,$67
          lqd       $23,3104($10)
          fma       $68,$34,$28,$68
          lqd       $24,3120($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#4
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,3328($10)
          fma       $66,$30,$22,$66
          lqd       $26,3344($10)
          fma       $67,$30,$23,$67
          lqd       $27,3360($10)
          fma       $68,$30,$24,$68
          lqd       $28,3376($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,3584($10)
          fma       $66,$34,$26,$66
          lqd       $22,3600($10)
          fma       $67,$34,$27,$67
          lqd       $23,3616($10)
          fma       $68,$34,$28,$68
          lqd       $24,3632($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,3840($10)
          fma       $62,$29,$22,$62
          lqd       $26,3856($10)
          fma       $63,$29,$23,$63
          lqd       $27,3872($10)
          fma       $64,$29,$24,$64
          lqd       $28,3888($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,64($9)
          fma       $62,$33,$26,$62
          lqd       $17,320($9)
          fma       $63,$33,$27,$63
          lqd       $18,576($9)
          fma       $64,$33,$28,$64
          lqd       $19,832($9)

          fma       $65,$34,$25,$65
          lqd       $21,4096($10)
          fma       $66,$34,$26,$66
          lqd       $22,4112($10)
          fma       $67,$34,$27,$67
          lqd       $23,4128($10)
          fma       $68,$34,$28,$68
          lqd       $24,4144($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#5
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,4352($10)
          fma       $66,$30,$22,$66
          lqd       $26,4368($10)
          fma       $67,$30,$23,$67
          lqd       $27,4384($10)
          fma       $68,$30,$24,$68
          lqd       $28,4400($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
```

```
        fma     $62,$33,$26,$62
        fma     $63,$33,$27,$63
        fma     $64,$33,$28,$64

        fma     $65,$34,$25,$65
        lqd     $21,4608($10)
        fma     $66,$34,$26,$66
        lqd     $22,4624($10)
        fma     $67,$34,$27,$67
        lqd     $23,4640($10)
        fma     $68,$34,$28,$68
        lqd     $24,4656($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$14
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$14
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$14
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$14

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76


        fma     $61,$29,$21,$61
        lqd     $25,4864($10)
        fma     $62,$29,$22,$62
        lqd     $26,4880($10)
        fma     $63,$29,$23,$63
        lqd     $27,4896($10)
        fma     $64,$29,$24,$64
        lqd     $28,4912($10)

        fma     $65,$30,$21,$65
        shufb   $33,$16,$16,$15
        fma     $66,$30,$22,$66
        shufb   $34,$17,$17,$15
        fma     $67,$30,$23,$67
        shufb   $35,$18,$18,$15
        fma     $68,$30,$24,$68
        shufb   $36,$19,$19,$15

        fma     $69,$31,$21,$69
        fma     $70,$31,$22,$70
        fma     $71,$31,$23,$71
        fma     $72,$31,$24,$72

        fma     $73,$32,$21,$73
        fma     $74,$32,$22,$74
        fma     $75,$32,$23,$75
        fma     $76,$32,$24,$76


        fma     $61,$33,$25,$61
        lqd     $16,80($9)
        fma     $62,$33,$26,$62
        lqd     $17,336($9)
        fma     $63,$33,$27,$63
        lqd     $18,592($9)
        fma     $64,$33,$28,$64
        lqd     $19,848($9)

        fma     $65,$34,$25,$65
        lqd     $21,5120($10)
        fma     $66,$34,$26,$66
        lqd     $22,5136($10)
        fma     $67,$34,$27,$67
        lqd     $23,5152($10)
        fma     $68,$34,$28,$68
        lqd     $24,5168($10)

        fma     $69,$35,$25,$69
        shufb   $29,$16,$16,$12
        fma     $70,$35,$26,$70
        shufb   $30,$17,$17,$12
        fma     $71,$35,$27,$71
        shufb   $31,$18,$18,$12
        fma     $72,$35,$28,$72
        shufb   $32,$19,$19,$12

        fma     $73,$36,$25,$73
        fma     $74,$36,$26,$74
        fma     $75,$36,$27,$75
        fma     $76,$36,$28,$76



#6
        fma     $61,$29,$21,$61
        fma     $62,$29,$22,$62
        fma     $63,$29,$23,$63
        fma     $64,$29,$24,$64

        fma     $65,$30,$21,$65
        lqd     $25,5376($10)
        fma     $66,$30,$22,$66
        lqd     $26,5392($10)
        fma     $67,$30,$23,$67
        lqd     $27,5408($10)
```

```
fma     $68,$30,$24,$68
lqd     $28,5424($10)

fma     $69,$31,$21,$69
shufb   $33,$16,$16,$13
fma     $70,$31,$22,$70
shufb   $34,$17,$17,$13
fma     $71,$31,$23,$71
shufb   $35,$18,$18,$13
fma     $72,$31,$24,$72
shufb   $36,$19,$19,$13

fma     $73,$32,$21,$73
fma     $74,$32,$22,$74
fma     $75,$32,$23,$75
fma     $76,$32,$24,$76

fma     $61,$33,$25,$61
fma     $62,$33,$26,$62
fma     $63,$33,$27,$63
fma     $64,$33,$28,$64

fma     $65,$34,$25,$65
lqd     $21,5632($10)
fma     $66,$34,$26,$66
lqd     $22,5648($10)
fma     $67,$34,$27,$67
lqd     $23,5664($10)
fma     $68,$34,$28,$68
lqd     $24,5680($10)

fma     $69,$35,$25,$69
shufb   $29,$16,$16,$14
fma     $70,$35,$26,$70
shufb   $30,$17,$17,$14
fma     $71,$35,$27,$71
shufb   $31,$18,$18,$14
fma     $72,$35,$28,$72
shufb   $32,$19,$19,$14

fma     $73,$36,$25,$73
fma     $74,$36,$26,$74
fma     $75,$36,$27,$75
fma     $76,$36,$28,$76


fma     $61,$29,$21,$61
lqd     $25,5888($10)
fma     $62,$29,$22,$62
lqd     $26,5904($10)
fma     $63,$29,$23,$63
lqd     $27,5920($10)
fma     $64,$29,$24,$64
lqd     $28,5936($10)

fma     $65,$30,$21,$65
shufb   $33,$16,$16,$15
fma     $66,$30,$22,$66
shufb   $34,$17,$17,$15
fma     $67,$30,$23,$67
shufb   $35,$18,$18,$15
fma     $68,$30,$24,$68
shufb   $36,$19,$19,$15

fma     $69,$31,$21,$69
fma     $70,$31,$22,$70
fma     $71,$31,$23,$71
fma     $72,$31,$24,$72

fma     $73,$32,$21,$73
fma     $74,$32,$22,$74
fma     $75,$32,$23,$75
fma     $76,$32,$24,$76


fma     $61,$33,$25,$61
lqd     $16,96($9)
fma     $62,$33,$26,$62
lqd     $17,352($9)
fma     $63,$33,$27,$63
lqd     $18,608($9)
fma     $64,$33,$28,$64
lqd     $19,864($9)

fma     $65,$34,$25,$65
lqd     $21,6144($10)
fma     $66,$34,$26,$66
lqd     $22,6160($10)
fma     $67,$34,$27,$67
lqd     $23,6176($10)
fma     $68,$34,$28,$68
lqd     $24,6192($10)

fma     $69,$35,$25,$69
shufb   $29,$16,$16,$12
fma     $70,$35,$26,$70
shufb   $30,$17,$17,$12
fma     $71,$35,$27,$71
shufb   $31,$18,$18,$12
fma     $72,$35,$28,$72
shufb   $32,$19,$19,$12

fma     $73,$36,$25,$73
```

```
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#7
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,6400($10)
          fma       $66,$30,$22,$66
          lqd       $26,6416($10)
          fma       $67,$30,$23,$67
          lqd       $27,6432($10)
          fma       $68,$30,$24,$68
          lqd       $28,6448($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,6656($10)
          fma       $66,$34,$26,$66
          lqd       $22,6672($10)
          fma       $67,$34,$27,$67
          lqd       $23,6688($10)
          fma       $68,$34,$28,$68
          lqd       $24,6704($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,6912($10)
          fma       $62,$29,$22,$62
          lqd       $26,6928($10)
          fma       $63,$29,$23,$63
          lqd       $27,6944($10)
          fma       $64,$29,$24,$64
          lqd       $28,6960($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,112($9)
          fma       $62,$33,$26,$62
          lqd       $17,368($9)
          fma       $63,$33,$27,$63
          lqd       $18,624($9)
          fma       $64,$33,$28,$64
          lqd       $19,880($9)

          fma       $65,$34,$25,$65
          lqd       $21,7168($10)
          fma       $66,$34,$26,$66
          lqd       $22,7184($10)
          fma       $67,$34,$27,$67
          lqd       $23,7200($10)
          fma       $68,$34,$28,$68
          lqd       $24,7216($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#8
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,7424($10)
          fma       $66,$30,$22,$66
          lqd       $26,7440($10)
          fma       $67,$30,$23,$67
          lqd       $27,7456($10)
          fma       $68,$30,$24,$68
          lqd       $28,7472($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,7680($10)
          fma       $66,$34,$26,$66
          lqd       $22,7696($10)
          fma       $67,$34,$27,$67
          lqd       $23,7712($10)
          fma       $68,$34,$28,$68
          lqd       $24,7728($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,7936($10)
          fma       $62,$29,$22,$62
          lqd       $26,7952($10)
          fma       $63,$29,$23,$63
          lqd       $27,7968($10)
          fma       $64,$29,$24,$64
          lqd       $28,7984($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
```

```
#         a         $10,$10,$78
          shufb     $10,$58,$58,$15
          fma       $72,$31,$24,$72
          lnop

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,128($9)
          fma       $62,$33,$26,$62
          lqd       $17,384($9)
          fma       $63,$33,$27,$63
          lqd       $18,640($9)
          fma       $64,$33,$28,$64
          lqd       $19,896($9)

          fma       $65,$34,$25,$65
          lqd       $21,0($10)
          fma       $66,$34,$26,$66
          lqd       $22,16($10)
          fma       $67,$34,$27,$67
          lqd       $23,32($10)
          fma       $68,$34,$28,$68
          lqd       $24,48($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#9
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,256($10)
          fma       $66,$30,$22,$66
          lqd       $26,272($10)
          fma       $67,$30,$23,$67
          lqd       $27,288($10)
          fma       $68,$30,$24,$68
          lqd       $28,304($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,512($10)
          fma       $66,$34,$26,$66
          lqd       $22,528($10)
          fma       $67,$34,$27,$67
          lqd       $23,544($10)
          fma       $68,$34,$28,$68
          lqd       $24,560($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76
          fma       $61,$29,$21,$61
          lqd       $25,768($10)
          fma       $62,$29,$22,$62
          lqd       $26,784($10)
          fma       $63,$29,$23,$63
          lqd       $27,800($10)
          fma       $64,$29,$24,$64
          lqd       $28,816($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,144($9)
          fma       $62,$33,$26,$62
          lqd       $17,400($9)
          fma       $63,$33,$27,$63
          lqd       $18,656($9)
          fma       $64,$33,$28,$64
          lqd       $19,912($9)

          fma       $65,$34,$25,$65
          lqd       $21,1024($10)
          fma       $66,$34,$26,$66
          lqd       $22,1040($10)
          fma       $67,$34,$27,$67
          lqd       $23,1056($10)
          fma       $68,$34,$28,$68
          lqd       $24,1072($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#10
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,1280($10)
          fma       $66,$30,$22,$66
          lqd       $26,1296($10)
          fma       $67,$30,$23,$67
          lqd       $27,1312($10)
          fma       $68,$30,$24,$68
          lqd       $28,1328($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,1536($10)
          fma       $66,$34,$26,$66
          lqd       $22,1552($10)
          fma       $67,$34,$27,$67
          lqd       $23,1568($10)
          fma       $68,$34,$28,$68
          lqd       $24,1584($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,1792($10)
          fma       $62,$29,$22,$62
          lqd       $26,1808($10)
          fma       $63,$29,$23,$63
          lqd       $27,1824($10)
          fma       $64,$29,$24,$64
          lqd       $28,1840($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,160($9)
          fma       $62,$33,$26,$62
          lqd       $17,416($9)
          fma       $63,$33,$27,$63
          lqd       $18,672($9)
          fma       $64,$33,$28,$64
          lqd       $19,928($9)

          fma       $65,$34,$25,$65
          lqd       $21,2048($10)
          fma       $66,$34,$26,$66
          lqd       $22,2064($10)
          fma       $67,$34,$27,$67
          lqd       $23,2080($10)
          fma       $68,$34,$28,$68
          lqd       $24,2096($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$12
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#11
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,2304($10)
          fma       $66,$30,$22,$66
          lqd       $26,2320($10)
          fma       $67,$30,$23,$67
          lqd       $27,2336($10)
          fma       $68,$30,$24,$68
          lqd       $28,2352($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb    $36,$19,$19,$13

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76

          fma      $61,$33,$25,$61
          fma      $62,$33,$26,$62
          fma      $63,$33,$27,$63
          fma      $64,$33,$28,$64

          fma      $65,$34,$25,$65
          lqd      $21,2560($10)
          fma      $66,$34,$26,$66
          lqd      $22,2576($10)
          fma      $67,$34,$27,$67
          lqd      $23,2592($10)
          fma      $68,$34,$28,$68
          lqd      $24,2608($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$14
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$14
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$14
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$14

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76


          fma      $61,$29,$21,$61
          lqd      $25,2816($10)
          fma      $62,$29,$22,$62
          lqd      $26,2832($10)
          fma      $63,$29,$23,$63
          lqd      $27,2848($10)
          fma      $64,$29,$24,$64
          lqd      $28,2864($10)

          fma      $65,$30,$21,$65
          shufb    $33,$16,$16,$15
          fma      $66,$30,$22,$66
          shufb    $34,$17,$17,$15
          fma      $67,$30,$23,$67
          shufb    $35,$18,$18,$15
          fma      $68,$30,$24,$68
          shufb    $36,$19,$19,$15

          fma      $69,$31,$21,$69
          fma      $70,$31,$22,$70
          fma      $71,$31,$23,$71
          fma      $72,$31,$24,$72

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76


          fma      $61,$33,$25,$61
          lqd      $16,176($9)
          fma      $62,$33,$26,$62
          lqd      $17,432($9)
          fma      $63,$33,$27,$63
          lqd      $18,688($9)
          fma      $64,$33,$28,$64
          lqd      $19,944($9)

          fma      $65,$34,$25,$65
          lqd      $21,3072($10)
          fma      $66,$34,$26,$66
          lqd      $22,3088($10)
          fma      $67,$34,$27,$67
          lqd      $23,3104($10)
          fma      $68,$34,$28,$68
          lqd      $24,3120($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$12
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$12
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$12
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$12

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76



#12
          fma      $61,$29,$21,$61
          fma      $62,$29,$22,$62
          fma      $63,$29,$23,$63
          fma      $64,$29,$24,$64

          fma      $65,$30,$21,$65
          lqd      $25,3328($10)
          fma      $66,$30,$22,$66
          lqd      $26,3344($10)
          fma      $67,$30,$23,$67
          lqd      $27,3360($10)
          fma      $68,$30,$24,$68
          lqd      $28,3376($10)

          fma      $69,$31,$21,$69
          shufb    $33,$16,$16,$13
          fma      $70,$31,$22,$70
          shufb    $34,$17,$17,$13
          fma      $71,$31,$23,$71
          shufb    $35,$18,$18,$13
          fma      $72,$31,$24,$72
          shufb    $36,$19,$19,$13

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76

          fma      $61,$33,$25,$61
          fma      $62,$33,$26,$62
          fma      $63,$33,$27,$63
          fma      $64,$33,$28,$64

          fma      $65,$34,$25,$65
          lqd      $21,3584($10)
          fma      $66,$34,$26,$66
          lqd      $22,3600($10)
          fma      $67,$34,$27,$67
          lqd      $23,3616($10)
          fma      $68,$34,$28,$68
          lqd      $24,3632($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$14
          fma      $70,$35,$26,$70
          shufb    $30,$17,$17,$14
          fma      $71,$35,$27,$71
          shufb    $31,$18,$18,$14
          fma      $72,$35,$28,$72
          shufb    $32,$19,$19,$14

          fma      $73,$36,$25,$73
          fma      $74,$36,$26,$74
          fma      $75,$36,$27,$75
          fma      $76,$36,$28,$76


          fma      $61,$29,$21,$61
          lqd      $25,3840($10)
          fma      $62,$29,$22,$62
          lqd      $26,3856($10)
          fma      $63,$29,$23,$63
          lqd      $27,3872($10)
          fma      $64,$29,$24,$64
          lqd      $28,3888($10)

          fma      $65,$30,$21,$65
          shufb    $33,$16,$16,$15
          fma      $66,$30,$22,$66
          shufb    $34,$17,$17,$15
          fma      $67,$30,$23,$67
          shufb    $35,$18,$18,$15
          fma      $68,$30,$24,$68
          shufb    $36,$19,$19,$15

          fma      $69,$31,$21,$69
          fma      $70,$31,$22,$70
          fma      $71,$31,$23,$71
          fma      $72,$31,$24,$72

          fma      $73,$32,$21,$73
          fma      $74,$32,$22,$74
          fma      $75,$32,$23,$75
          fma      $76,$32,$24,$76


          fma      $61,$33,$25,$61
          lqd      $16,192($9)
          fma      $62,$33,$26,$62
          lqd      $17,448($9)
          fma      $63,$33,$27,$63
          lqd      $18,704($9)
          fma      $64,$33,$28,$64
          lqd      $19,960($9)

          fma      $65,$34,$25,$65
          lqd      $21,4096($10)
          fma      $66,$34,$26,$66
          lqd      $22,4112($10)
          fma      $67,$34,$27,$67
          lqd      $23,4128($10)
          fma      $68,$34,$28,$68
          lqd      $24,4144($10)

          fma      $69,$35,$25,$69
          shufb    $29,$16,$16,$12
```

```
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$12
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$12
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$12

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76



#13
          fma       $61,$29,$21,$61
          fma       $62,$29,$22,$62
          fma       $63,$29,$23,$63
          fma       $64,$29,$24,$64

          fma       $65,$30,$21,$65
          lqd       $25,4352($10)
          fma       $66,$30,$22,$66
          lqd       $26,4368($10)
          fma       $67,$30,$23,$67
          lqd       $27,4384($10)
          fma       $68,$30,$24,$68
          lqd       $28,4400($10)

          fma       $69,$31,$21,$69
          shufb     $33,$16,$16,$13
          fma       $70,$31,$22,$70
          shufb     $34,$17,$17,$13
          fma       $71,$31,$23,$71
          shufb     $35,$18,$18,$13
          fma       $72,$31,$24,$72
          shufb     $36,$19,$19,$13

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76

          fma       $61,$33,$25,$61
          fma       $62,$33,$26,$62
          fma       $63,$33,$27,$63
          fma       $64,$33,$28,$64

          fma       $65,$34,$25,$65
          lqd       $21,4608($10)
          fma       $66,$34,$26,$66
          lqd       $22,4624($10)
          fma       $67,$34,$27,$67
          lqd       $23,4640($10)
          fma       $68,$34,$28,$68
          lqd       $24,4656($10)

          fma       $69,$35,$25,$69
          shufb     $29,$16,$16,$14
          fma       $70,$35,$26,$70
          shufb     $30,$17,$17,$14
          fma       $71,$35,$27,$71
          shufb     $31,$18,$18,$14
          fma       $72,$35,$28,$72
          shufb     $32,$19,$19,$14

          fma       $73,$36,$25,$73
          fma       $74,$36,$26,$74
          fma       $75,$36,$27,$75
          fma       $76,$36,$28,$76


          fma       $61,$29,$21,$61
          lqd       $25,4864($10)
          fma       $62,$29,$22,$62
          lqd       $26,4880($10)
          fma       $63,$29,$23,$63
          lqd       $27,4896($10)
          fma       $64,$29,$24,$64
          lqd       $28,4912($10)

          fma       $65,$30,$21,$65
          shufb     $33,$16,$16,$15
          fma       $66,$30,$22,$66
          shufb     $34,$17,$17,$15
          fma       $67,$30,$23,$67
          shufb     $35,$18,$18,$15
          fma       $68,$30,$24,$68
          shufb     $36,$19,$19,$15

          fma       $69,$31,$21,$69
          fma       $70,$31,$22,$70
          fma       $71,$31,$23,$71
          fma       $72,$31,$24,$72

          fma       $73,$32,$21,$73
          fma       $74,$32,$22,$74
          fma       $75,$32,$23,$75
          fma       $76,$32,$24,$76


          fma       $61,$33,$25,$61
          lqd       $16,208($9)
```

```
            fma      $62,$33,$26,$62
            lqd      $17,464($9)
            fma      $63,$33,$27,$63
            lqd      $18,720($9)
            fma      $64,$33,$28,$64
            lqd      $19,976($9)

            fma      $65,$34,$25,$65
            lqd      $21,5120($10)
            fma      $66,$34,$26,$66
            lqd      $22,5136($10)
            fma      $67,$34,$27,$67
            lqd      $23,5152($10)
            fma      $68,$34,$28,$68
            lqd      $24,5168($10)

            fma      $69,$35,$25,$69
            shufb    $29,$16,$16,$12
            fma      $70,$35,$26,$70
            shufb    $30,$17,$17,$12
            fma      $71,$35,$27,$71
            shufb    $31,$18,$18,$12
            fma      $72,$35,$28,$72
            shufb    $32,$19,$19,$12

            fma      $73,$36,$25,$73
            fma      $74,$36,$26,$74
            fma      $75,$36,$27,$75
            fma      $76,$36,$28,$76



#14
            fma      $61,$29,$21,$61
            fma      $62,$29,$22,$62
            fma      $63,$29,$23,$63
            fma      $64,$29,$24,$64

            fma      $65,$30,$21,$65
            lqd      $25,5376($10)
            fma      $66,$30,$22,$66
            lqd      $26,5392($10)
            fma      $67,$30,$23,$67
            lqd      $27,5408($10)
            fma      $68,$30,$24,$68
            lqd      $28,5424($10)

            fma      $69,$31,$21,$69
            shufb    $33,$16,$16,$13
            fma      $70,$31,$22,$70
            shufb    $34,$17,$17,$13
            fma      $71,$31,$23,$71
            shufb    $35,$18,$18,$13
            fma      $72,$31,$24,$72
            shufb    $36,$19,$19,$13

            fma      $73,$32,$21,$73
            fma      $74,$32,$22,$74
            fma      $75,$32,$23,$75
            fma      $76,$32,$24,$76

            fma      $61,$33,$25,$61
            fma      $62,$33,$26,$62
            fma      $63,$33,$27,$63
            fma      $64,$33,$28,$64

            fma      $65,$34,$25,$65
            lqd      $21,5632($10)
            fma      $66,$34,$26,$66
            lqd      $22,5648($10)
            fma      $67,$34,$27,$67
            lqd      $23,5664($10)
            fma      $68,$34,$28,$68
            lqd      $24,5680($10)

            fma      $69,$35,$25,$69
            shufb    $29,$16,$16,$14
            fma      $70,$35,$26,$70
            shufb    $30,$17,$17,$14
            fma      $71,$35,$27,$71
            shufb    $31,$18,$18,$14
            fma      $72,$35,$28,$72
            shufb    $32,$19,$19,$14

            fma      $73,$36,$25,$73
            fma      $74,$36,$26,$74
            fma      $75,$36,$27,$75
            fma      $76,$36,$28,$76


            fma      $61,$29,$21,$61
            lqd      $25,5888($10)
            fma      $62,$29,$22,$62
            lqd      $26,5904($10)
            fma      $63,$29,$23,$63
            lqd      $27,5920($10)
            fma      $64,$29,$24,$64
            lqd      $28,5936($10)

            fma      $65,$30,$21,$65
            shufb    $33,$16,$16,$15
            fma      $66,$30,$22,$66
            shufb    $34,$17,$17,$15
```

```
        fma      $67,$30,$23,$67
        shufb    $35,$18,$18,$15
        fma      $68,$30,$24,$68
        shufb    $36,$19,$19,$15

        fma      $69,$31,$21,$69
        fma      $70,$31,$22,$70
        fma      $71,$31,$23,$71
        fma      $72,$31,$24,$72

        fma      $73,$32,$21,$73
        fma      $74,$32,$22,$74
        fma      $75,$32,$23,$75
        fma      $76,$32,$24,$76


        fma      $61,$33,$25,$61
        lqd      $16,224($9)
        fma      $62,$33,$26,$62
        lqd      $17,480($9)
        fma      $63,$33,$27,$63
        lqd      $18,736($9)
        fma      $64,$33,$28,$64
        lqd      $19,992($9)

        fma      $65,$34,$25,$65
        lqd      $21,6144($10)
        fma      $66,$34,$26,$66
        lqd      $22,6160($10)
        fma      $67,$34,$27,$67
        lqd      $23,6176($10)
        fma      $68,$34,$28,$68
        lqd      $24,6192($10)

        fma      $69,$35,$25,$69
        shufb    $29,$16,$16,$12
        fma      $70,$35,$26,$70
        shufb    $30,$17,$17,$12
        fma      $71,$35,$27,$71
        shufb    $31,$18,$18,$12
        fma      $72,$35,$28,$72
        shufb    $32,$19,$19,$12

        fma      $73,$36,$25,$73
        fma      $74,$36,$26,$74
        fma      $75,$36,$27,$75
        fma      $76,$36,$28,$76



#15
        fma      $61,$29,$21,$61
        fma      $62,$29,$22,$62
        fma      $63,$29,$23,$63
        fma      $64,$29,$24,$64

        fma      $65,$30,$21,$65
        lqd      $25,6400($10)
        fma      $66,$30,$22,$66
        lqd      $26,6416($10)
        fma      $67,$30,$23,$67
        lqd      $27,6432($10)
        fma      $68,$30,$24,$68
        lqd      $28,6448($10)

        fma      $69,$31,$21,$69
        shufb    $33,$16,$16,$13
        fma      $70,$31,$22,$70
        shufb    $34,$17,$17,$13
        fma      $71,$31,$23,$71
        shufb    $35,$18,$18,$13
        fma      $72,$31,$24,$72
        shufb    $36,$19,$19,$13

        fma      $73,$32,$21,$73
        fma      $74,$32,$22,$74
        fma      $75,$32,$23,$75
        fma      $76,$32,$24,$76

        fma      $61,$33,$25,$61
        fma      $62,$33,$26,$62
        fma      $63,$33,$27,$63
        fma      $64,$33,$28,$64

        fma      $65,$34,$25,$65
        lqd      $21,6656($10)
        fma      $66,$34,$26,$66
        lqd      $22,6672($10)
        fma      $67,$34,$27,$67
        lqd      $23,6688($10)
        fma      $68,$34,$28,$68
        lqd      $24,6704($10)

        fma      $69,$35,$25,$69
        shufb    $29,$16,$16,$14
        fma      $70,$35,$26,$70
        shufb    $30,$17,$17,$14
        fma      $71,$35,$27,$71
        shufb    $31,$18,$18,$14
        fma      $72,$35,$28,$72
        shufb    $32,$19,$19,$14

        fma      $73,$36,$25,$73
```

```
            fma       $74,$36,$26,$74
            fma       $75,$36,$27,$75
            fma       $76,$36,$28,$76


            fma       $61,$29,$21,$61
            lqd       $25,6912($10)
            fma       $62,$29,$22,$62
            lqd       $26,6928($10)
            fma       $63,$29,$23,$63
            lqd       $27,6944($10)
            fma       $64,$29,$24,$64
            lqd       $28,6960($10)

            fma       $65,$30,$21,$65
            shufb     $33,$16,$16,$15
            fma       $66,$30,$22,$66
            shufb     $34,$17,$17,$15
            fma       $67,$30,$23,$67
            shufb     $35,$18,$18,$15
            fma       $68,$30,$24,$68
            shufb     $36,$19,$19,$15

            fma       $69,$31,$21,$69
            fma       $70,$31,$22,$70
            fma       $71,$31,$23,$71
            fma       $72,$31,$24,$72

            fma       $73,$32,$21,$73
            fma       $74,$32,$22,$74
            fma       $75,$32,$23,$75
            fma       $76,$32,$24,$76


            fma       $61,$33,$25,$61
            lqd       $16,240($9)
            fma       $62,$33,$26,$62
            lqd       $17,496($9)
            fma       $63,$33,$27,$63
            lqd       $18,752($9)
            fma       $64,$33,$28,$64
            lqd       $19,1008($9)

            fma       $65,$34,$25,$65
            lqd       $21,7168($10)
            fma       $66,$34,$26,$66
            lqd       $22,7184($10)
            fma       $67,$34,$27,$67
            lqd       $23,7200($10)
            fma       $68,$34,$28,$68
            lqd       $24,7216($10)

            fma       $69,$35,$25,$69
            shufb     $29,$16,$16,$12
            fma       $70,$35,$26,$70
            shufb     $30,$17,$17,$12
            fma       $71,$35,$27,$71
            shufb     $31,$18,$18,$12
            fma       $72,$35,$28,$72
            shufb     $32,$19,$19,$12

            fma       $73,$36,$25,$73
            fma       $74,$36,$26,$74
            fma       $75,$36,$27,$75
            fma       $76,$36,$28,$76


#16
            fma       $61,$29,$21,$61
            fma       $62,$29,$22,$62
            fma       $63,$29,$23,$63
            fma       $64,$29,$24,$64

            fma       $65,$30,$21,$65
            lqd       $25,7424($10)
            fma       $66,$30,$22,$66
            lqd       $26,7440($10)
            fma       $67,$30,$23,$67
            lqd       $27,7456($10)
            fma       $68,$30,$24,$68
            lqd       $28,7472($10)

            fma       $69,$31,$21,$69
            shufb     $33,$16,$16,$13
            fma       $70,$31,$22,$70
            shufb     $34,$17,$17,$13
            fma       $71,$31,$23,$71
            shufb     $35,$18,$18,$13
            fma       $72,$31,$24,$72
            shufb     $36,$19,$19,$13

            fma       $73,$32,$21,$73
            shufb     $29,$16,$16,$14
            fma       $74,$32,$22,$74
            shufb     $30,$17,$17,$14
            fma       $75,$32,$23,$75
            shufb     $31,$18,$18,$14
            fma       $76,$32,$24,$76
            shufb     $32,$19,$19,$14

            fma       $61,$33,$25,$61
            fma       $62,$33,$26,$62
            fma       $63,$33,$27,$63
            fma       $64,$33,$28,$64

            fma       $65,$34,$25,$65
            lqd       $21,7680($10)
            fma       $66,$34,$26,$66
            lqd       $22,7696($10)
            fma       $67,$34,$27,$67
            lqd       $23,7712($10)
            fma       $68,$34,$28,$68
            lqd       $24,7728($10)

            fma       $69,$35,$25,$69
            cgti      $57,$54,0
            fma       $70,$35,$26,$70
            lnop
            fma       $71,$35,$27,$71
            fma       $72,$35,$28,$72

            fma       $73,$36,$25,$73
            lqd       $25,7936($10)
            fma       $74,$36,$26,$74
            lqd       $26,7952($10)
            fma       $75,$36,$27,$75
            lqd       $27,7968($10)
            fma       $76,$36,$28,$76
            lqd       $28,7984($10)

            fma       $61,$29,$21,$61
            shufb     $33,$16,$16,$15
            fma       $62,$29,$22,$62
            shufb     $34,$17,$17,$15
            fma       $63,$29,$23,$63
            shufb     $35,$18,$18,$15
            fma       $64,$29,$24,$64
            shufb     $36,$19,$19,$15

            fma       $65,$30,$21,$65
            a         $9,$59,$56
            fma       $66,$30,$22,$66
            lnop
            fma       $67,$30,$23,$67
            rotqbyi   $10,$4,0
            fma       $68,$30,$24,$68
            rotqbyi   $59,$9,0

            fma       $69,$31,$21,$69
            lqd       $16,0($9)
            fma       $70,$31,$22,$70
            lqd       $17,256($9)
            fma       $71,$31,$23,$71
            lqd       $18,512($9)
            fma       $72,$31,$24,$72
            lqd       $19,768($9)

            fma       $73,$32,$21,$73
            lqd       $21,0($10)
            fma       $74,$32,$22,$74
            lqd       $22,16($10)
            fma       $75,$32,$23,$75
            lqd       $23,32($10)
            fma       $76,$32,$24,$76
            lqd       $24,48($10)

            fma       $61,$33,$25,$61
            shufb     $29,$16,$16,$12
            fma       $62,$33,$26,$62
            shufb     $30,$17,$17,$12
            fma       $63,$33,$27,$63
            shufb     $31,$18,$18,$12
            fma       $64,$33,$28,$64
            shufb     $32,$19,$19,$12

            fma       $65,$34,$25,$65
            fma       $66,$34,$26,$66
            fma       $67,$34,$27,$67
            stqd      $61,0($77)
            fma       $68,$34,$28,$68
            stqd      $62,16($77)

            fma       $69,$35,$25,$69
            stqd      $63,32($77)
            fma       $70,$35,$26,$70
            stqd      $64,48($77)
            fma       $71,$35,$27,$71
            stqd      $65,256($77)
            fma       $72,$35,$28,$72
            stqd      $66,272($77)

            fma       $73,$36,$25,$73
            stqd      $67,288($77)
            fma       $74,$36,$26,$74
            stqd      $68,304($77)
            fma       $75,$36,$27,$75
            stqd      $69,512($77)
            fma       $76,$36,$28,$76
            stqd      $70,528($77)



            brnz      $57,.LOOP

            stqd      $71,544($77)
            stqd      $72,560($77)
```

```
          stqd      $73,768($77)
          stqd      $74,784($77)
          stqd      $75,800($77)
          stqd      $76,816($77)

.L3:
          bi        $lr
          .size     matmul_SIMD64, .-matmul_SIMD64
          .ident    "Handcoded 64x64 matmul for CELL BE"
```